1.28 Gbps firmware
(or: Maria does VHDL Pt 3)

Maria Mironova, Timon Heim

Weekly student instrumentation meeting
09/06/2023

BERKELEY LAB

ATLAS EXPERIMENT

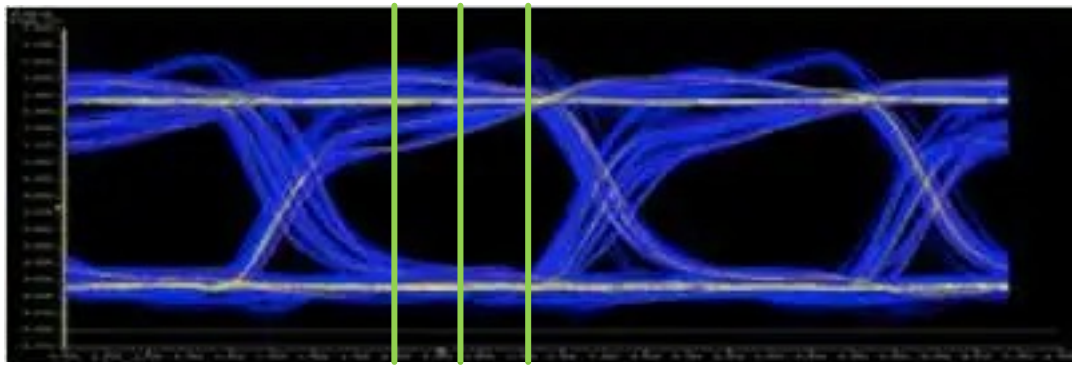
Motivation

- ITk requires readout link speed of 1.28 Gbits/s to cope with the high data rates, especially in the innermost regions
- We need to be able to test data transmission for QC
- Currently most of the testing in YARR is done at 640 Mbps, and readout at 1.28 Gbps was not possible (reliably)
- Goals:
  - Make 1.28 Gbps firmware work and understand better the behaviour of the data transmission
  - Figure out how to reliably run at 1.28 Gpbs, which settings work best, and how to implement a stable solution into YARR
  - Include a data transmission test in module QC

Recap from last time

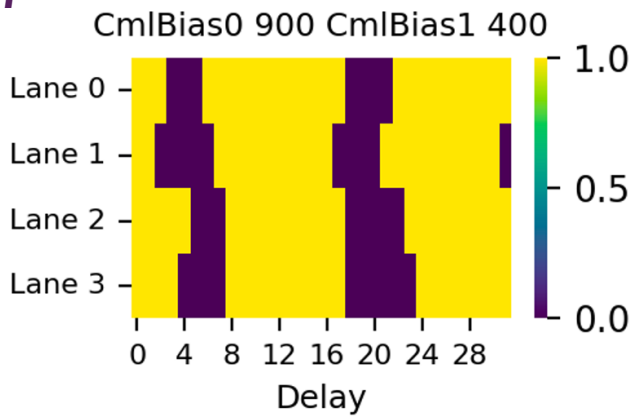
- Need to find a good delay setting at which we sample our data
- This was done in a smart way in the original YARR FW, where the FW would identify the transition regions and then decide on a good setting → did not work reliably
- Implemented a new deserialiser where one can change the delay setting
- Was able to scan over all delay settings and make a very basic eye diagram by checking whether the RX link is synchronised
- Could find settings at which a digital scan would run without errors (on a Trenz card), though not reliably
- Results could not be reproduced on an Xpress7 card

Eye diagram



Different delay settings

Basic "eye diagram" from RX sync



Error counter

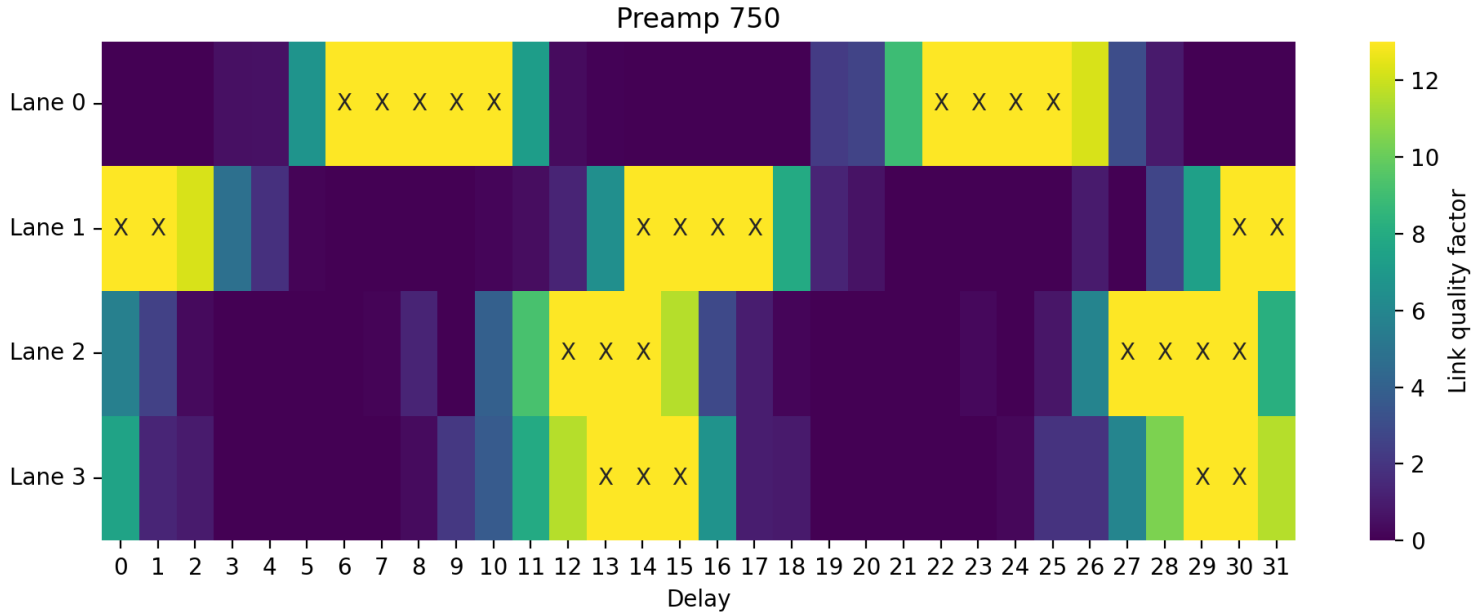
- Proper error counter implemented by Francesco
  - → Counting the "idles" seen
  - → We continuously get "idles" from the service blocks
  - → Can check whether we get the expected number, if not we have data transmission errors
  - → Can also now make a more sophisticated eye diagram

- Error counter should always be 204208, due to the number of service blocks sent
- Define **link quality factor** as:

$$Link\ quality = \log\left(\frac{1}{\frac{|count - 204208|}{204208}}\right)$$

→ Goes to infinity at perfect transmission

- Perfect link quality indicated by "X"



Part I: Testing FPGA settings

- Readout speed:
  - Trenz card specifications only go up to 1.2 Gbps, while we run at 1.28 Gbps
  - Test switching to 1.2 Gbps
- Differential termination of data lanes enabled/disabled
- VMUX settings:
  - Danny previously observed potential issues with data transmissions in the serial power chain (link)
  - Caused by noise on the pre-regulator reference voltage, which gets worse with larger shunt current
  - Could be improved by setting the VMUX monitoring setting to the pre-regulator reference voltage, which adds a capacitance that reduces the noise
  - Test VMUX setting 63 (open) and 32 (Vref_pre)
- FPGA operational voltage (1.8 V vs 2.5 V)

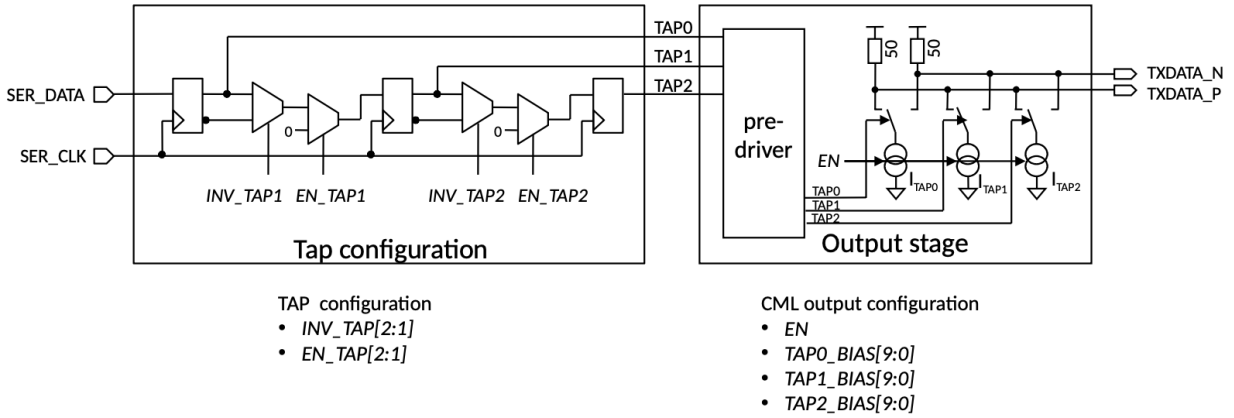
Setting	Option 1	Option 2
Voltage	1.8 V	2.5 V
VMUX	32	63
Differential Termination	no	yes
Readout speed	1.28 Gbps	1.2 Gbps

Cml bias scan

- For these measurements, we also need to consider various permutations of CmlBias settings

Reminder:

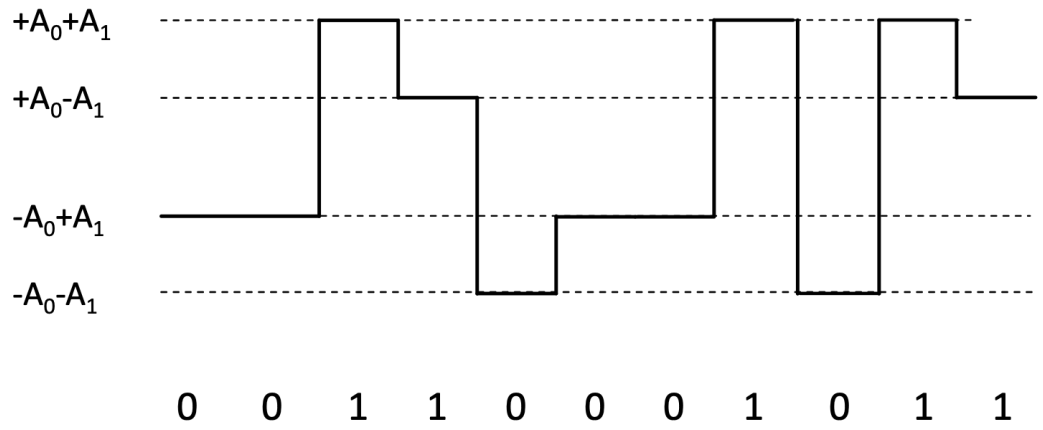
- In addition to just measuring the eye diagrams, ITkPix has parameters for signal shaping → **e.g. CmlBias1 for adding pre-emphasis**
- Good settings should decrease the time it takes to transition in the eye diagram and **increase eye opening width**
- Scan over CmlBias0 and CmlBias1 to understand what good settings would be



TAP configuration
- *INV_TAP[2:1]*
- *EN_TAP[2:1]*

CML output configuration
- *EN*
- *TAP0_BIAS[9:0]*
- *TAP1_BIAS[9:0]*
- *TAP2_BIAS[9:0]*

2-TAP pre-emphasis (main-post)

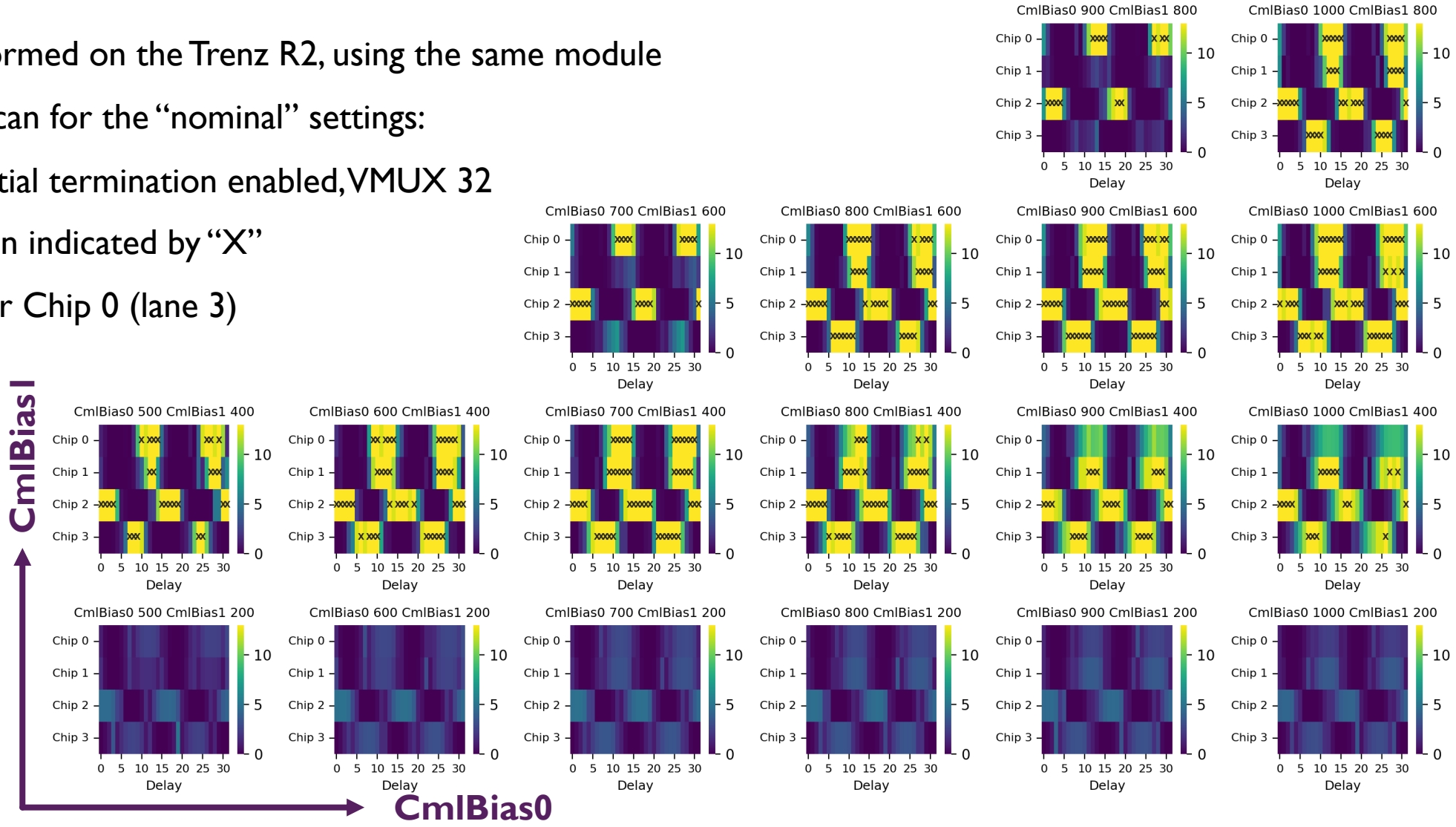


1.8 V, 1.28 Gbps, DiffTerm enabled, VMUX 32

- All measurements performed on the Trenz R2, using the same module
- Example of a CmlBias scan for the "nominal" settings:

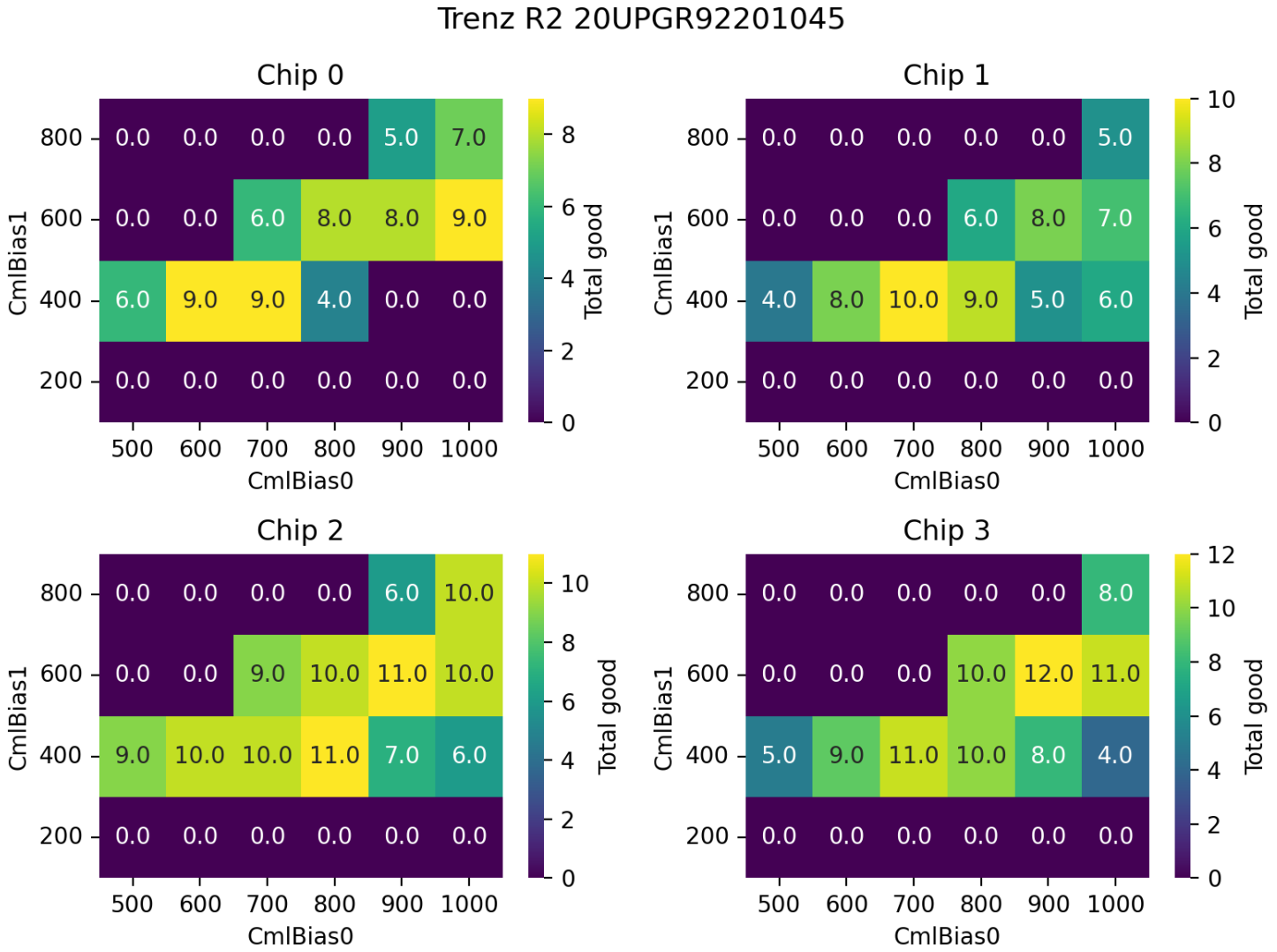
→ 1.8 V, 1.28 Gbps, differential termination enabled, VMUX 32

- Perfect data transmission indicated by "X"
- Interesting behaviour for Chip 0 (lane 3)



1.8 V, 1.28 Gbps, DiffTerm enabled, VMUX 32

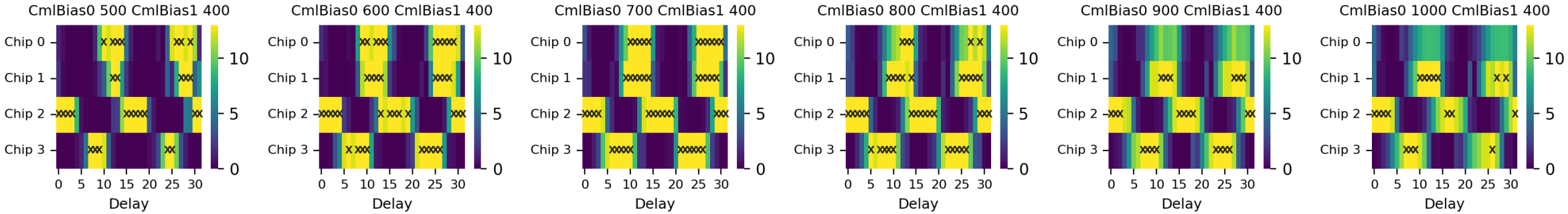
- Count the number of perfect delay settings in a window of two clock periods (28 delay settings) → will refer to this as "eye width" throughout
- Larger number means more margin for good data transmission
- Preferred CmlBias settings vary quite a lot for this setup



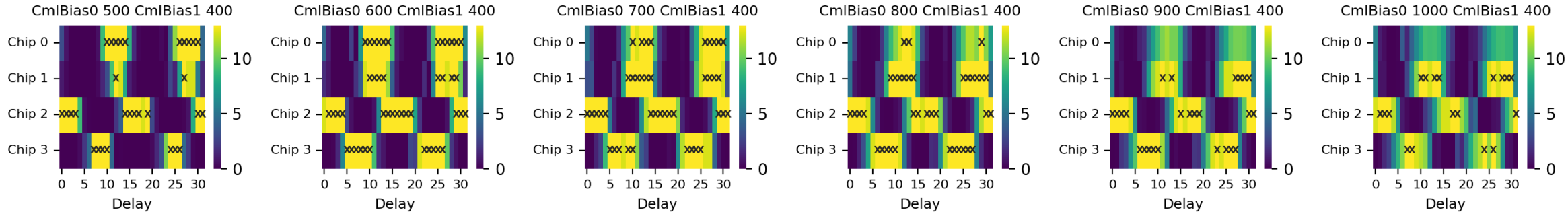
1.8 V vs 2.5 V

- Start testing different FPGA settings, starting with FPGA voltage 1.8 V vs 2.5 V:

1.8 V



2.5 V



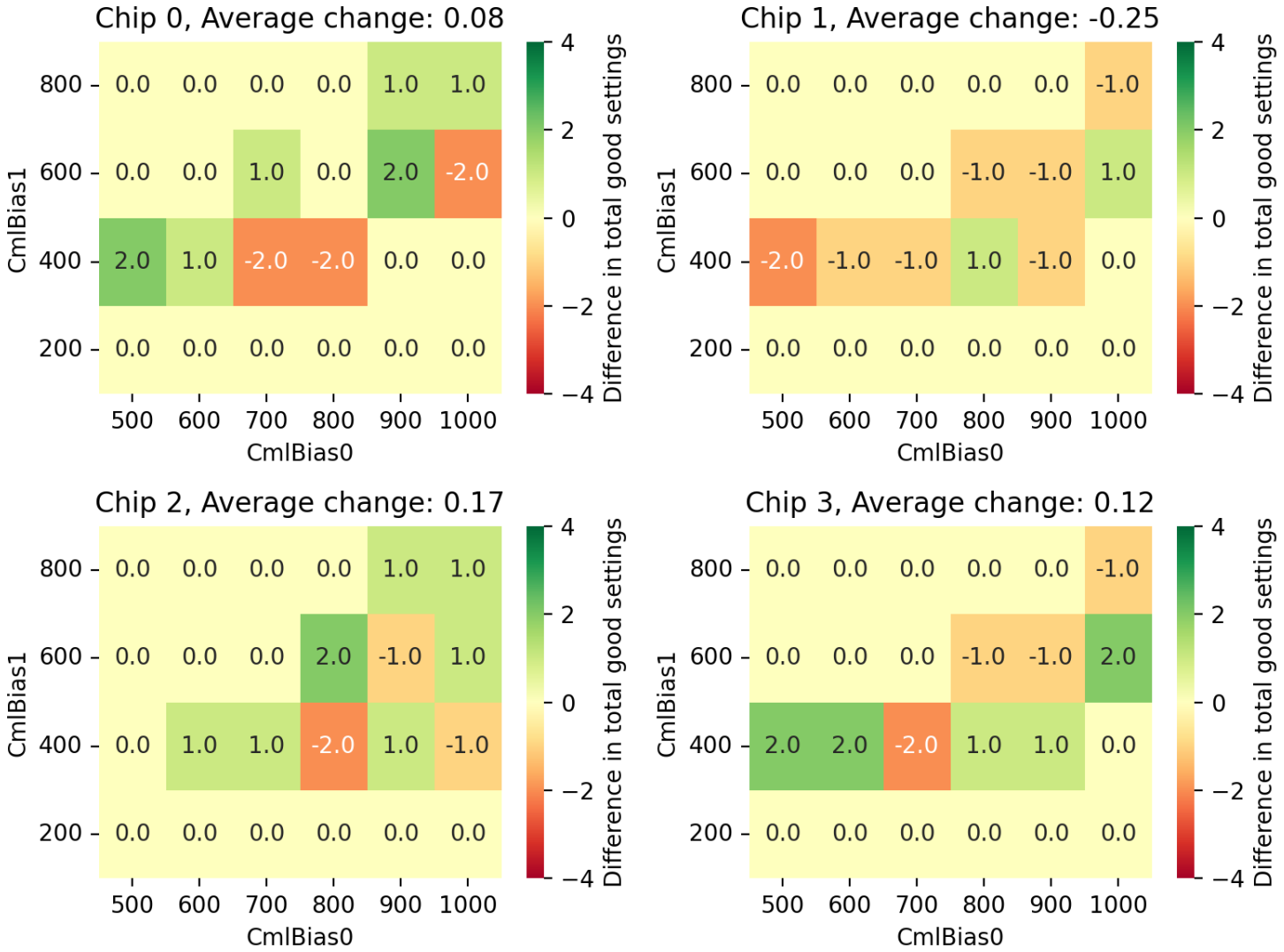
→ No noticeable difference by eye

1.8 V vs 2.5 V FPGA voltage

- Quantify the impact of the changes by taking the difference in eye widths
- Red means the new setting is worse, green means the new setting is better
- Also calculate average change for each chip

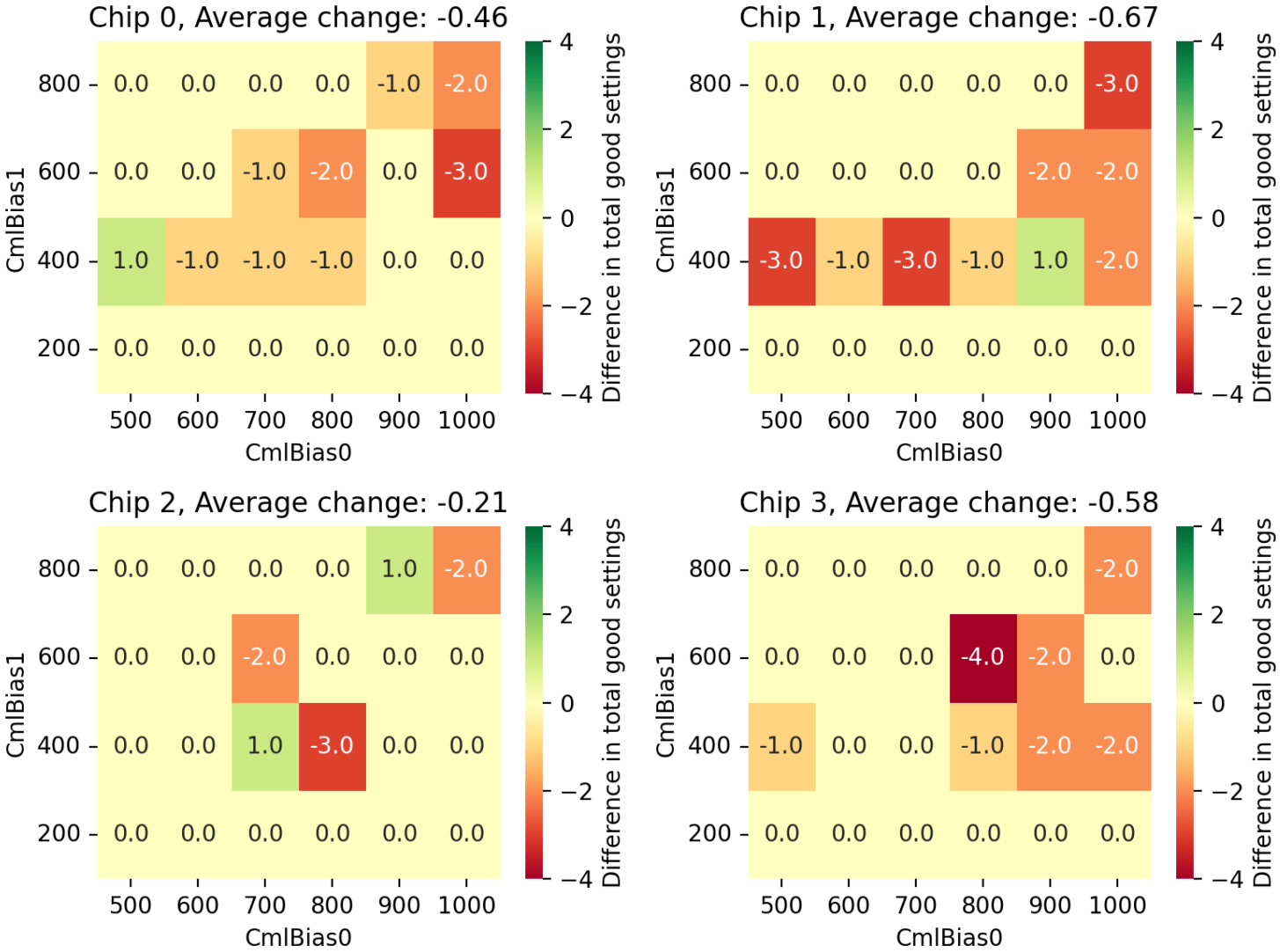
→ Impact of going to 2.5 V is very small, though overall slightly positive

→ Though probably not worth doing as we've always been operating at 1.8 V



VMUX 32 vs VMUX 63

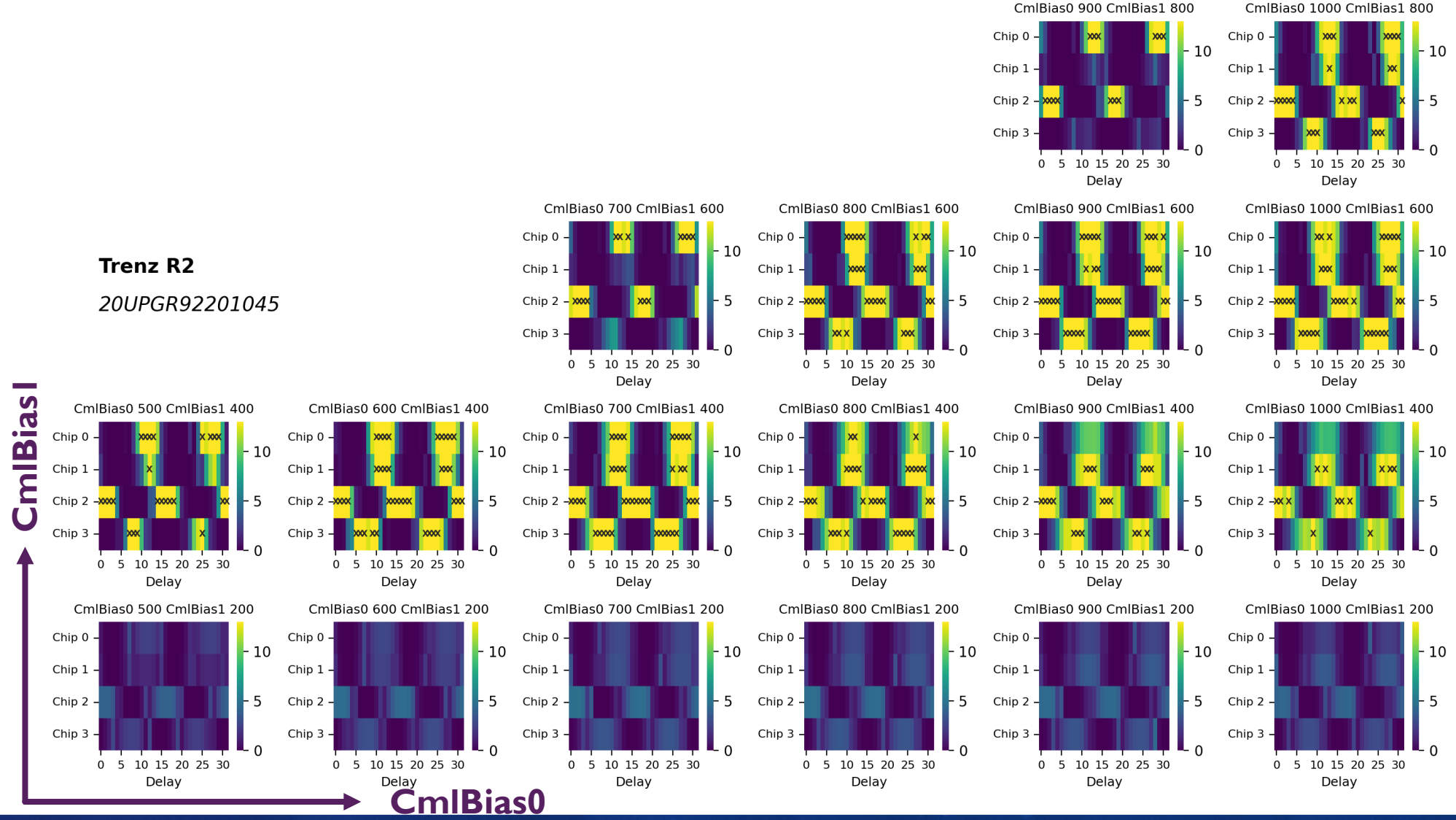
- Set VMUX to be open, rater than set to Vref_pre
- Reminder: VMUX 32 was chosen to reduce noise on pre-regulator voltage
- Switching back to VMUX 63 does reduce the eye width
- Setting VMUX 32 would probably make module QC more complicated, is it worth doing?



VMUX 32 vs VMUX 63

Trenz R2

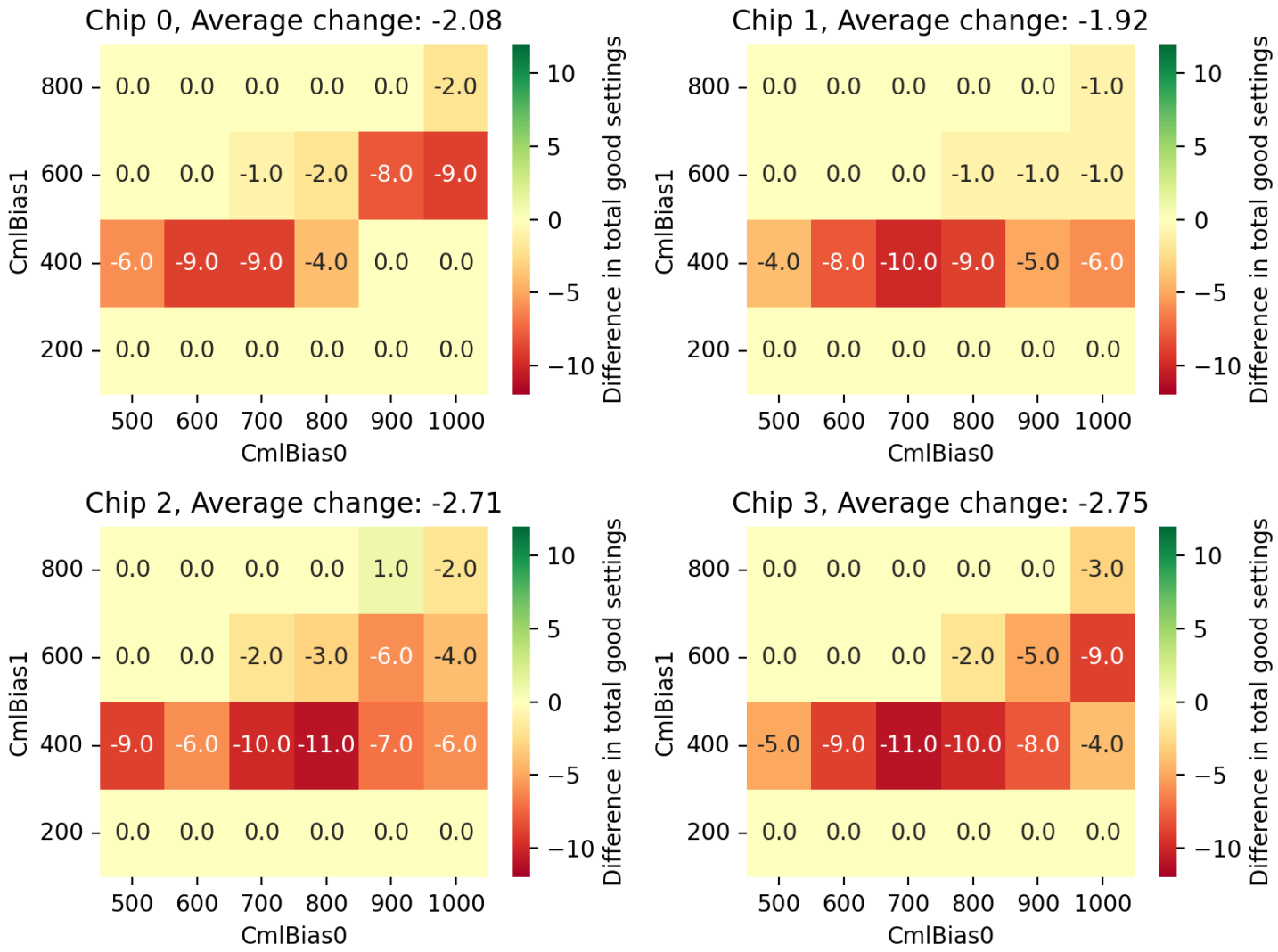
20UPGR92201045



Differential termination enabled/disabled

- Tried disabling differential termination
- Huge impact, make data transmission quality a lot worse, as expected

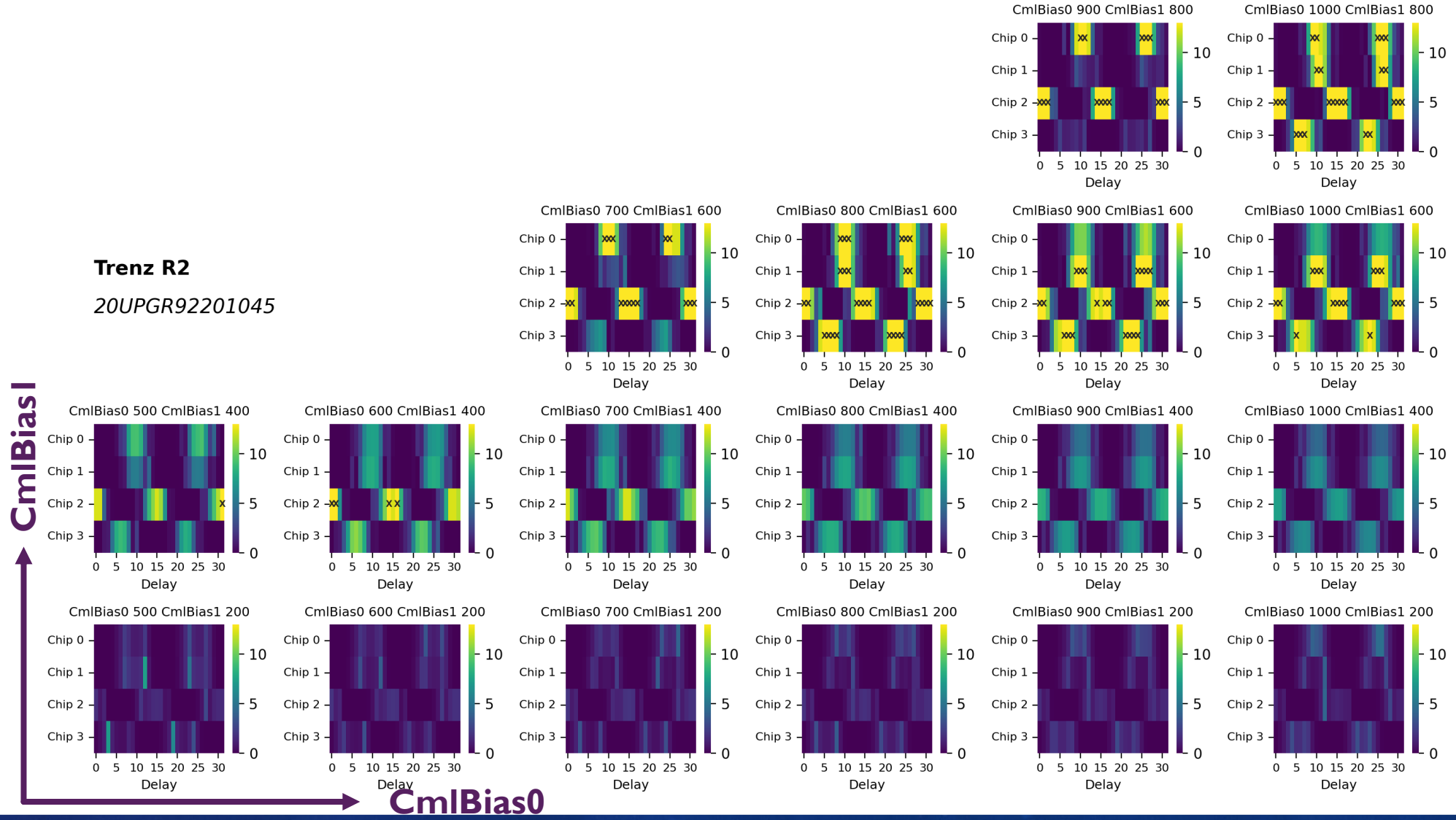
→ Go with differential termination enabled



Differential termination enabled/disabled

Trenz R2

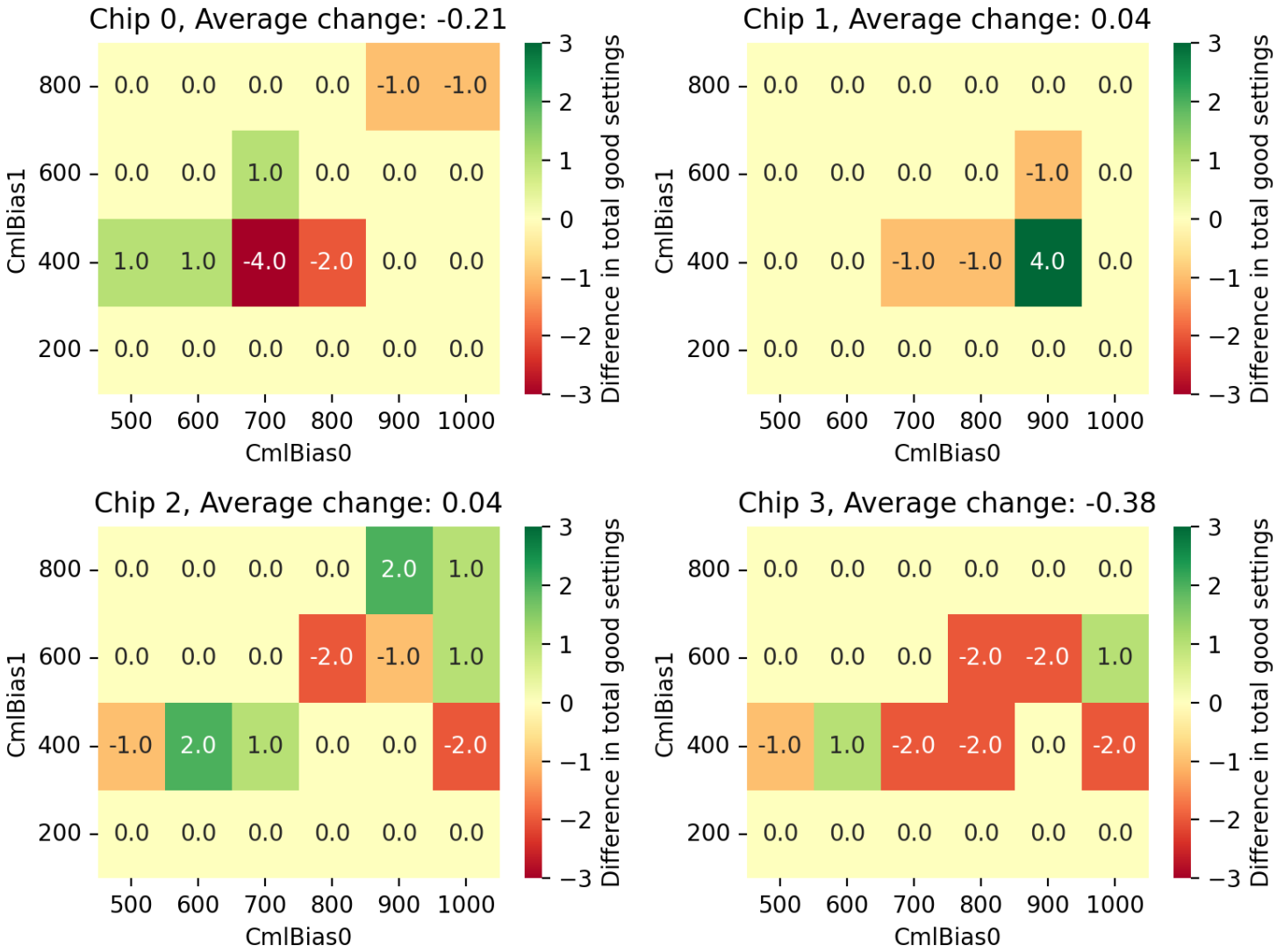
20UPGR92201045



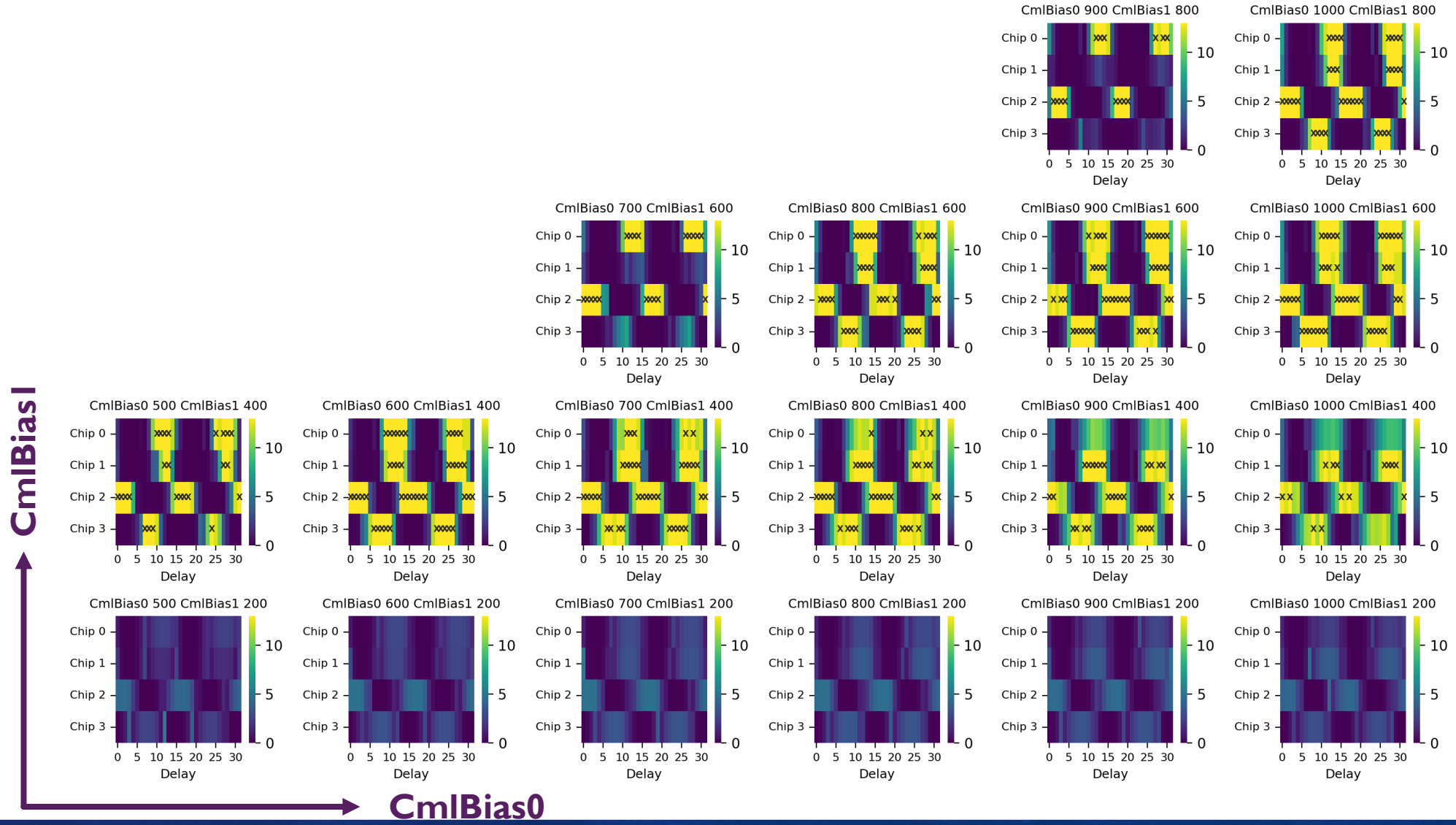
Readout speed: 1.28 Gbps vs 1.2 Gbps

- Tried going from 1.28 Gbps to 1.2 Gbps
- Some changes for some of the settings, but overall negligible effect

→ Go with 1.28 Gbps



Readout speed: 1.28 Gbps vs 1.2 Gbps



Summary Part I

- Tested various FPGA setting to try and find optimal configuration
- What should we do about VMUX?

- To Do: Test all of this again on an Xpress7 card

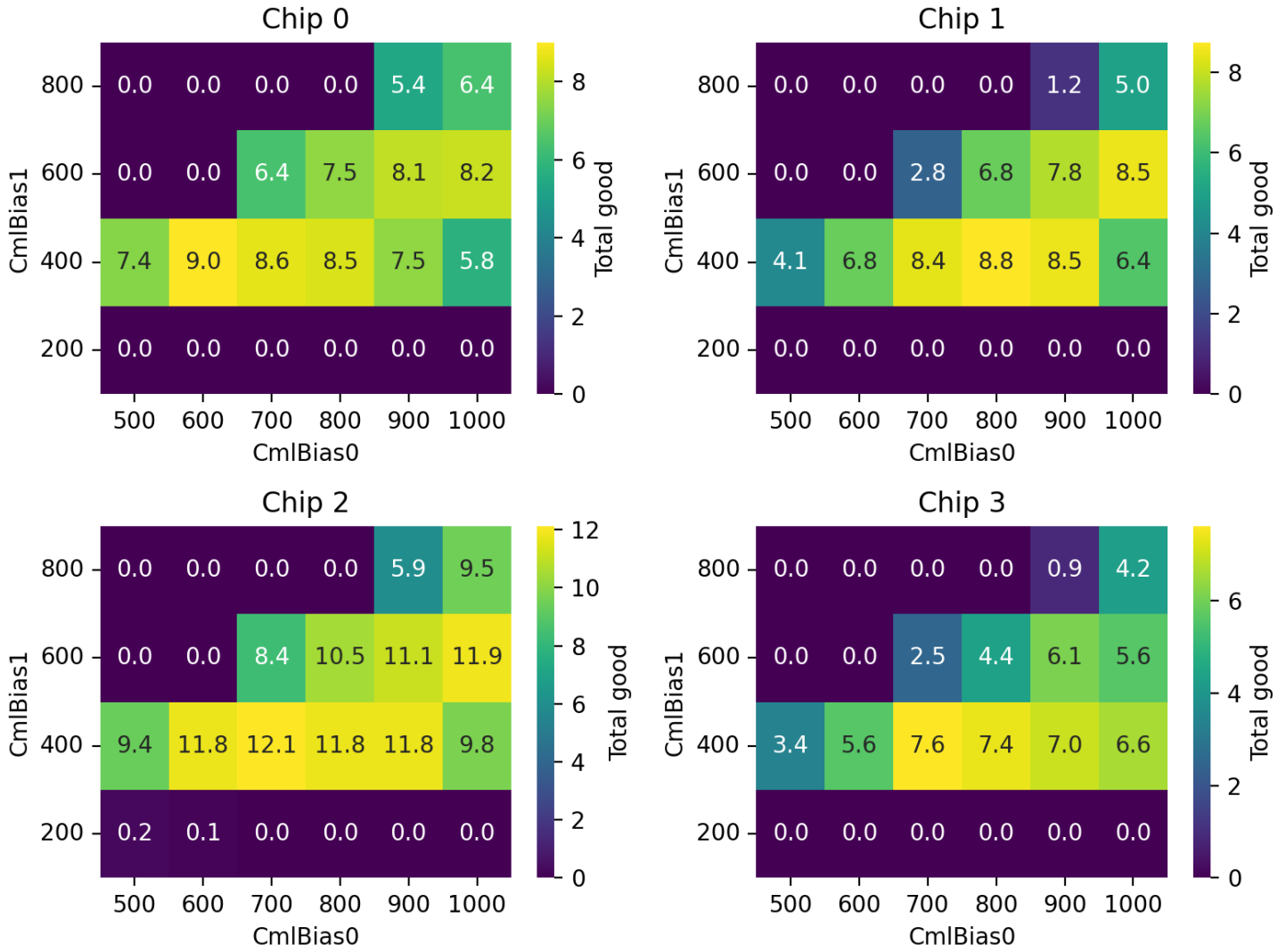
Setting	Option 1	Option 2
Voltage	**1.8 V**	2.5 V
VMUX	32	63
Differential Termination	no	**yes**
Readout speed	**1.28 Gbps**	1.2 Gbps

Part 2: Systematic study

- In QC we are considering to just hardcode the delay setting into the controller config → need to understand the variability of settings between different setups
- Prelimiary proposal for module QC: Run eye diagram scan and write delay settings to the controller config → need to figure out how often this has to be done
- Need to consider:
  - Different modules
  - Different ports on the same FPGA
  - Different kinds of FPGA (Trenz R1, Trenz R2, XpressK7)
- Study performed on Trenz R1 with the nominal configuration (1.8 V, VMUX 32, DiffTerm enabled, 1.28 Gbps)

Aside: CmlBias settings

- Took the average of the CmlBias scans of all of the measurements on the Trenz R1 card to find the best CmlBias settings
- Observe some chip-dependence in the ideal settings:
  - Chip 0: 600/400
  - Chip 1: 800/400
  - Chip 2: 700/400
  - Chip 3: 700/400
- Needs some further investigation…

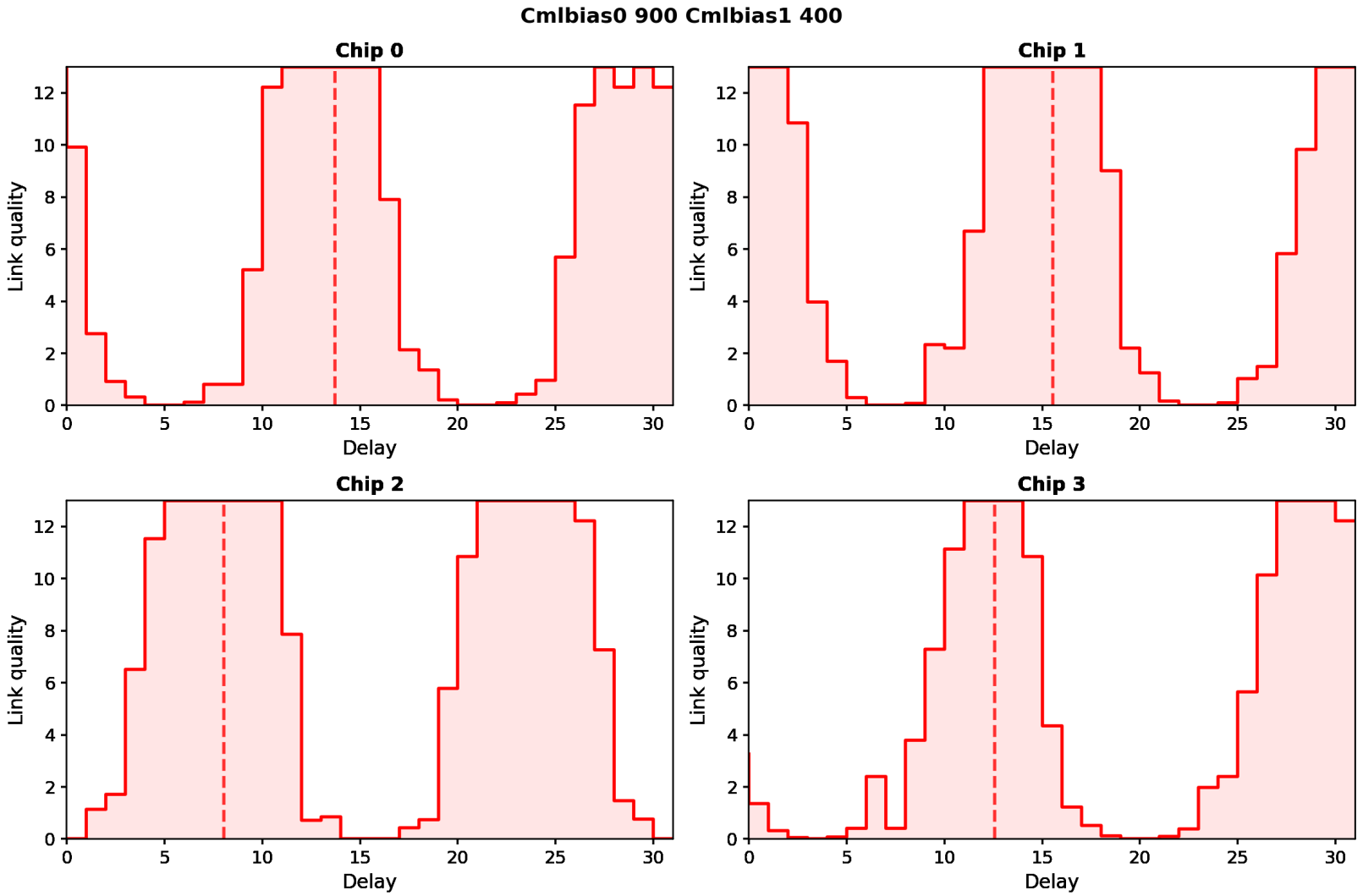


Example plot

- Compare different setups at the same CmlBias settings
- Plot link quality as a function as a function of delay:

$$Link\ quality = \log\left(\frac{1}{\frac{|count-204208|}{204208}}\right)$$

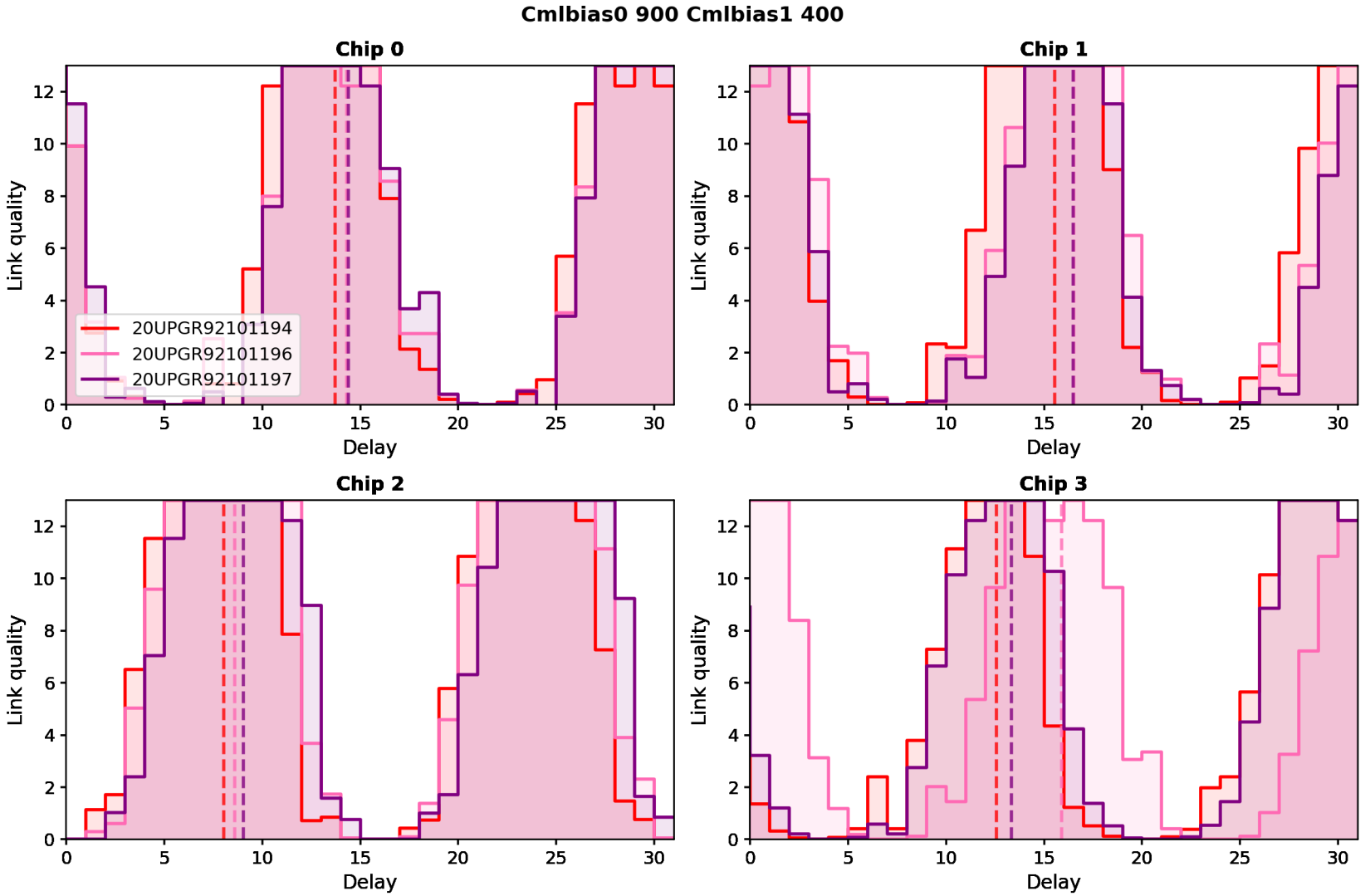
- Perfect data transmission when link quality is 13
- Dashed line indicates middle of eye



Comparison of different modules

- Compare eye diagrams of different quad modules
- For three quad modules build around the same time we see consistent behaviour
- Can run all three modules at the same delay settings
- **Communication is very reliable and I can run many (O(10)) digital scans without any errors**

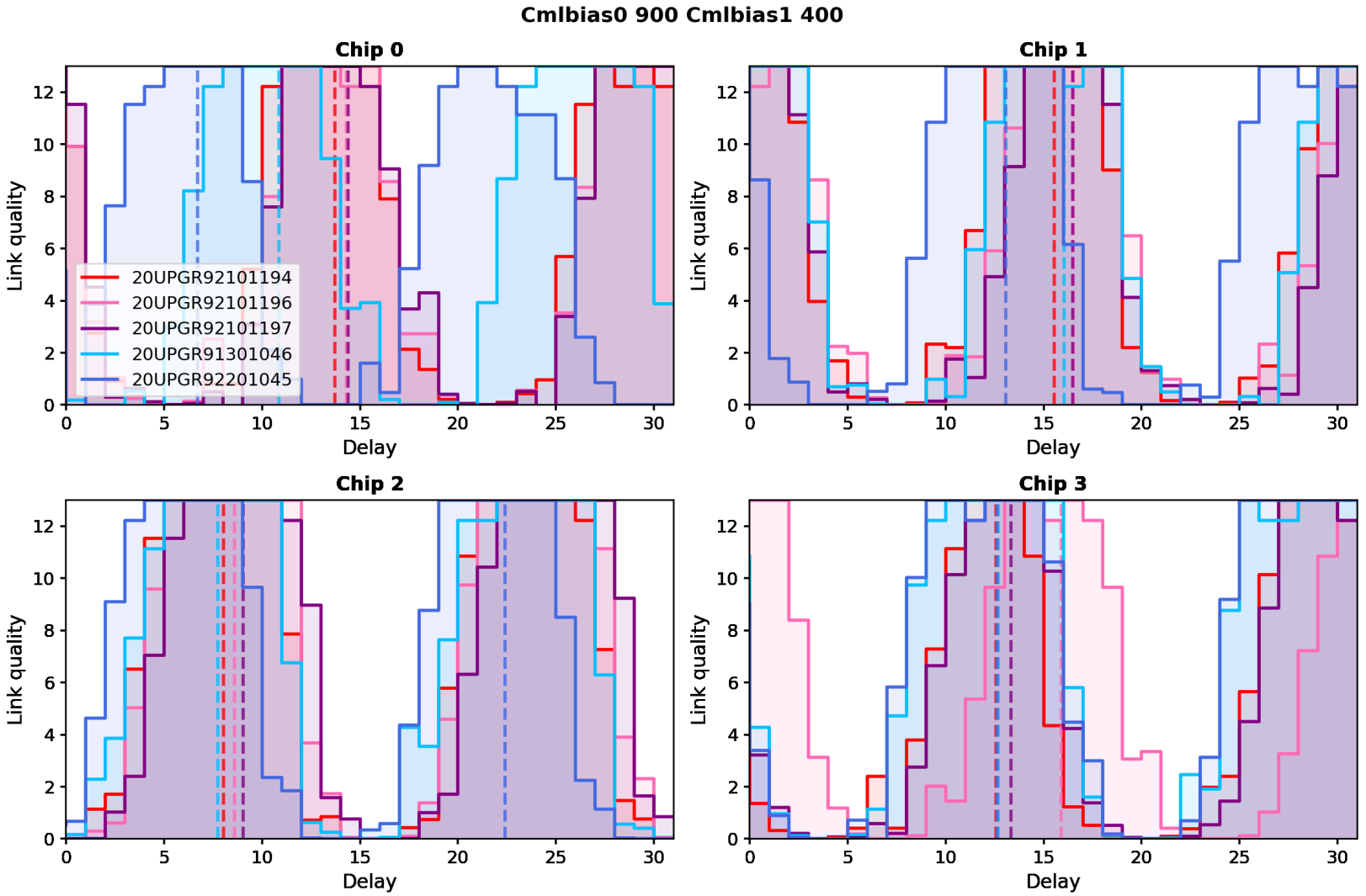
→ To Do: quantify reliability for longer periods



Comparison of different modules

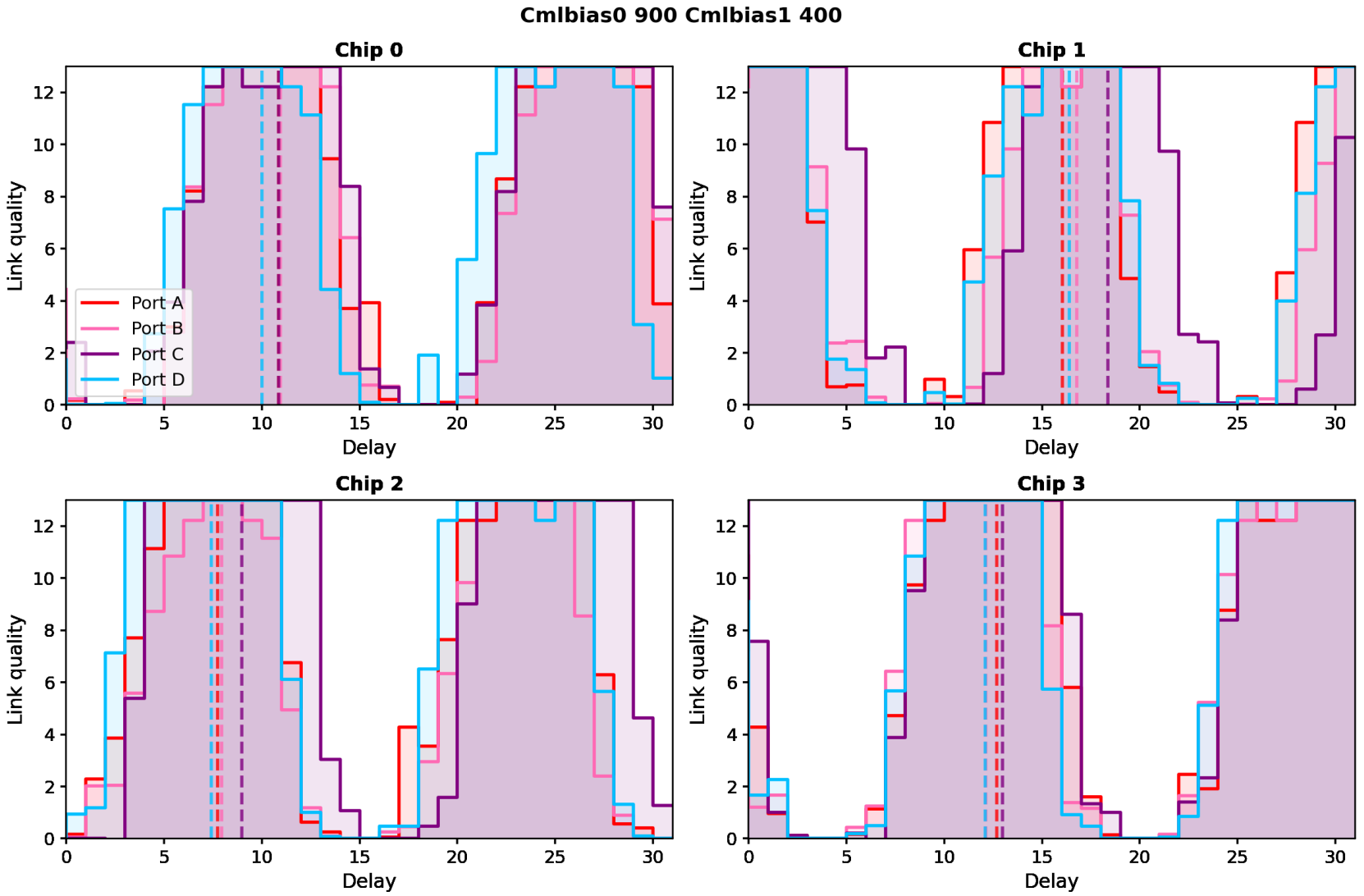
- But, some modules also show differences
- **045** and **046** have the same flex version, but from a different manufacturer

→ This means we would likely want to run the eye diagram scan for each module



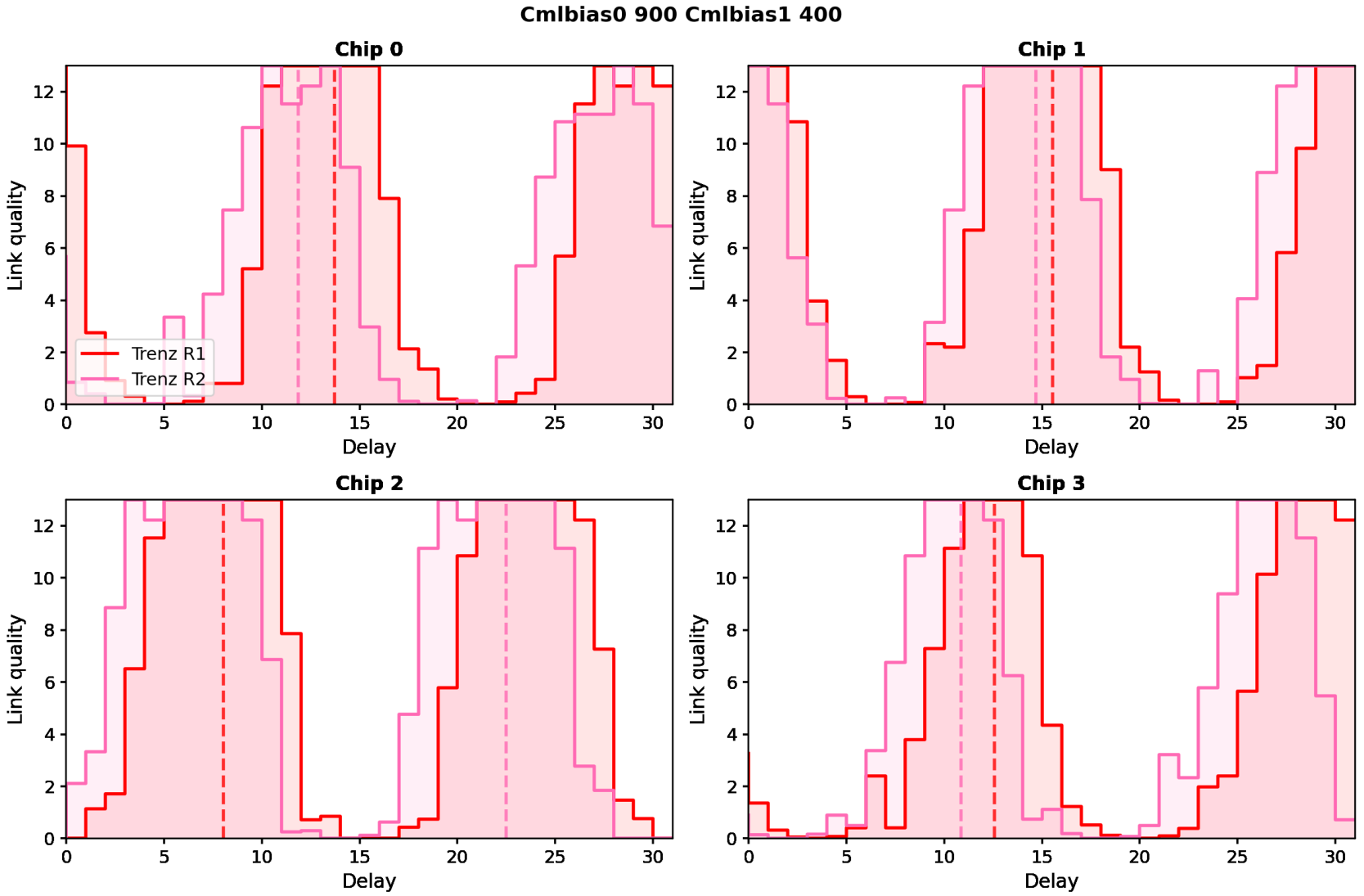
Comparison of different FPGA ports

- Compare different ports on the Trenz card
- Find consistent delay settings, though some ports seem to work better than others



Comparison of different FPGA cards

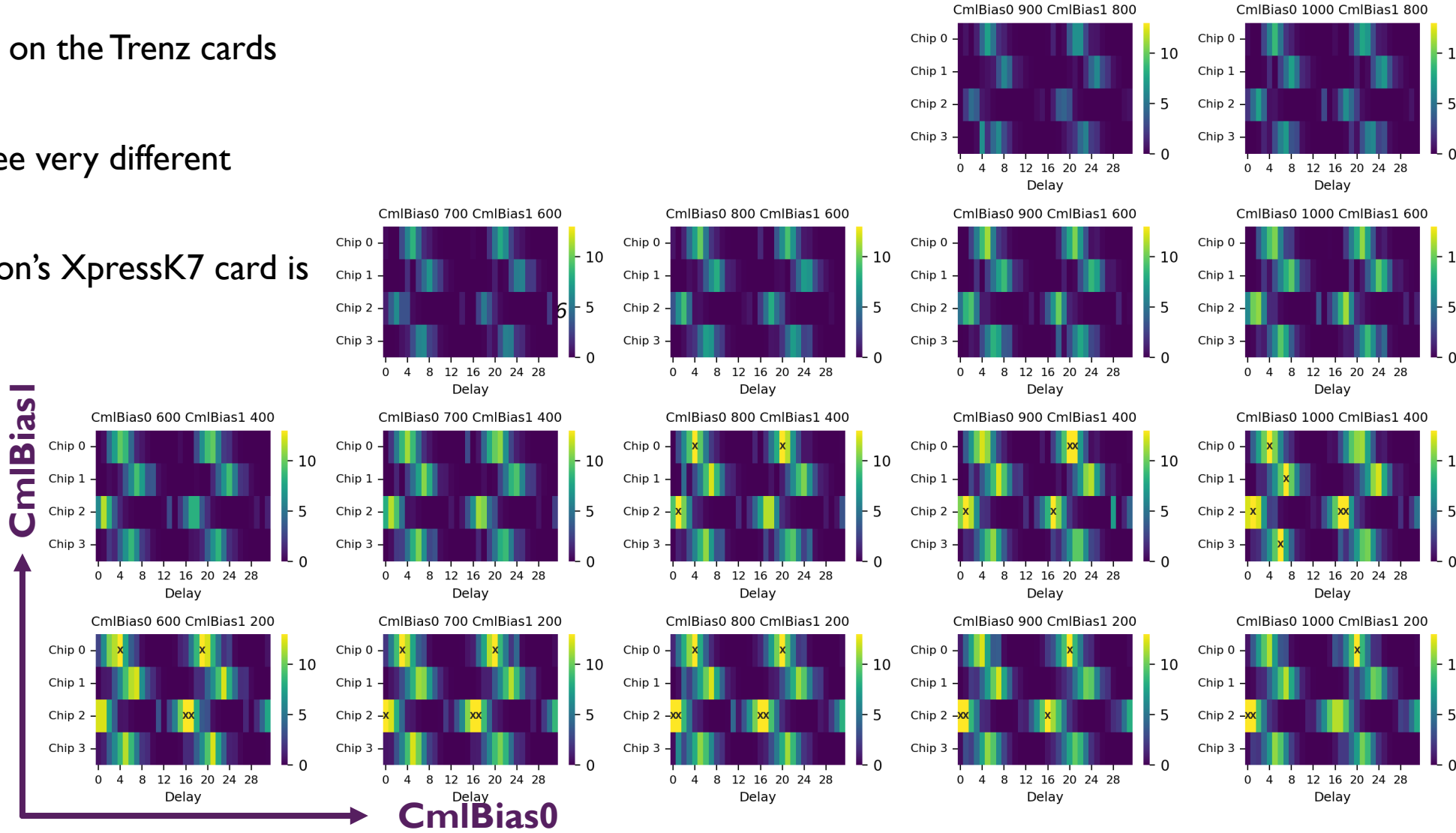
- Compare different FPGA cards: Trenz R1 and R2
- Optimal delay is pretty similar
- Trenz R2 performs a lot worse
- To Do: Try to see if the difference improves with adding some constraints to the delay in the firmware



FPGA dependence: XpressK7

- Operation is pretty stable on the Trenz cards now
- On my XpressK7 card I see very different behaviour
- Data transmission on Timon's XpressK7 card is relatively stable

→ To be investigated further



Summary

- Data transmission is complicated
- With the new firmware and error counter we can get stable data transmission, which should give us a way forward for module QC
- Proposal for module QC:
  - For each module, before the first test at 1.28 Gbps, run an eye diagram scan, which writes delay settings to the controller config,
  - Should also include the capabilities for running CmlBias scans, though this doesn't need to be included in the QC procedure
- Open questions:
  - Optimal CmlBias settings → need to gather some more statistics on different FPGAs/modules
  - Understand better the hardware dependence - why does my XpressK7 card look so much worse than the Trenz and Timon's XpressK7?

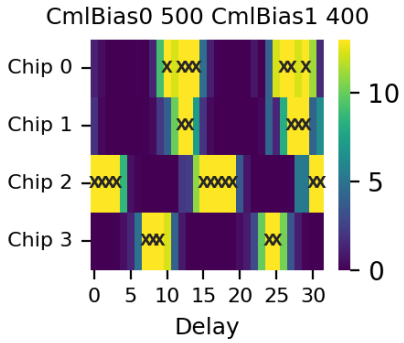
Thank you!

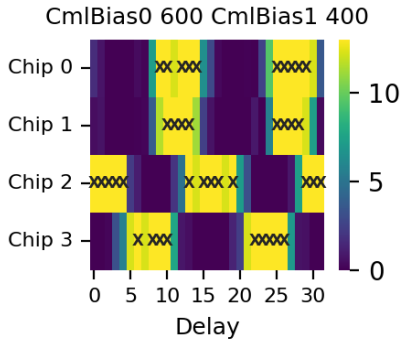
Questions?

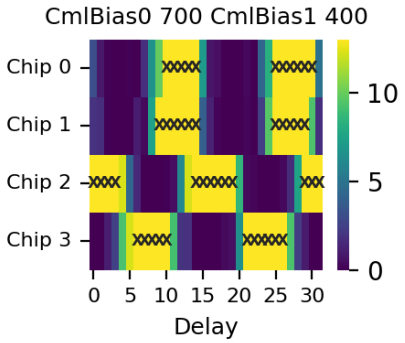
Nominal: 1.8 V, 1.28 Gbps, DiffTerm, VMUX 32

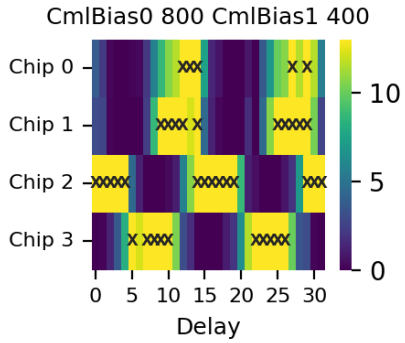
Trenz R2

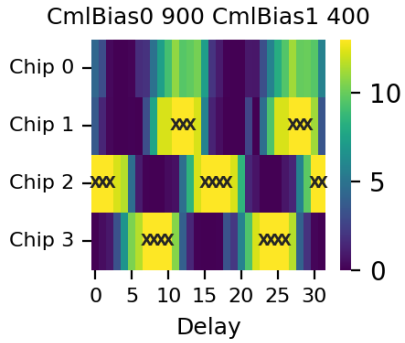
20UPGR92201045

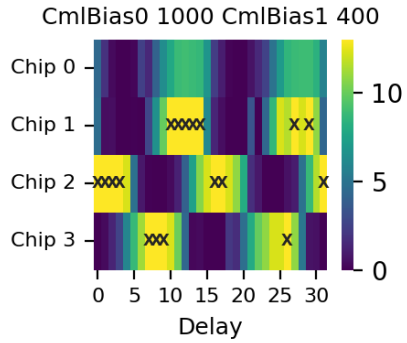


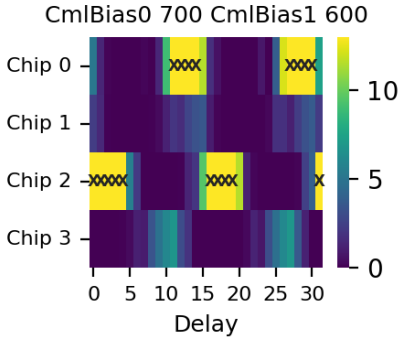


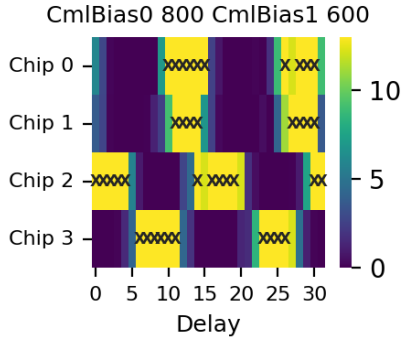


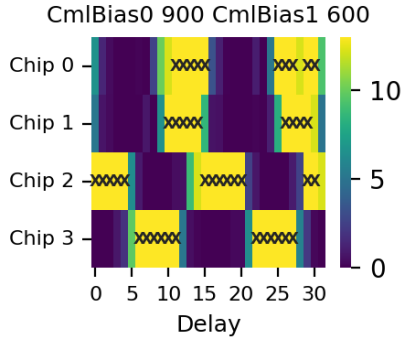


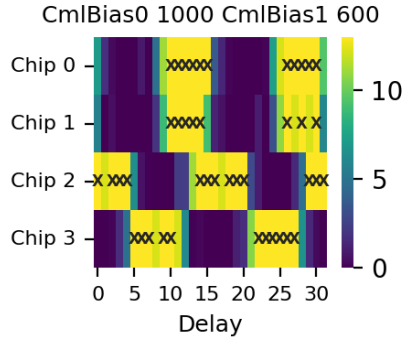


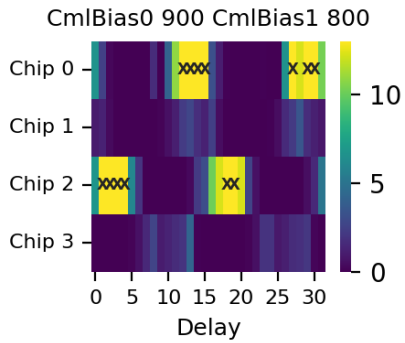


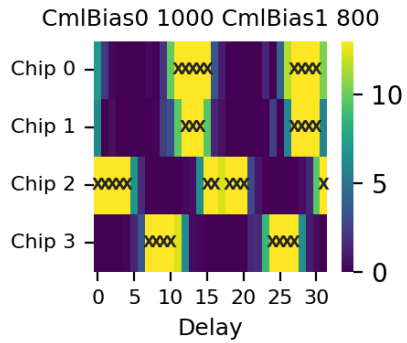








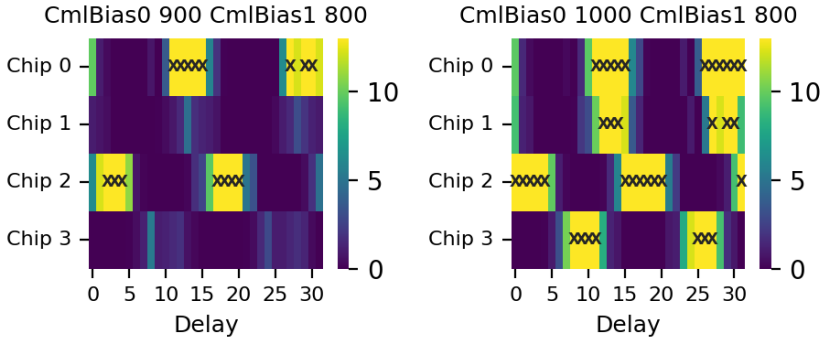


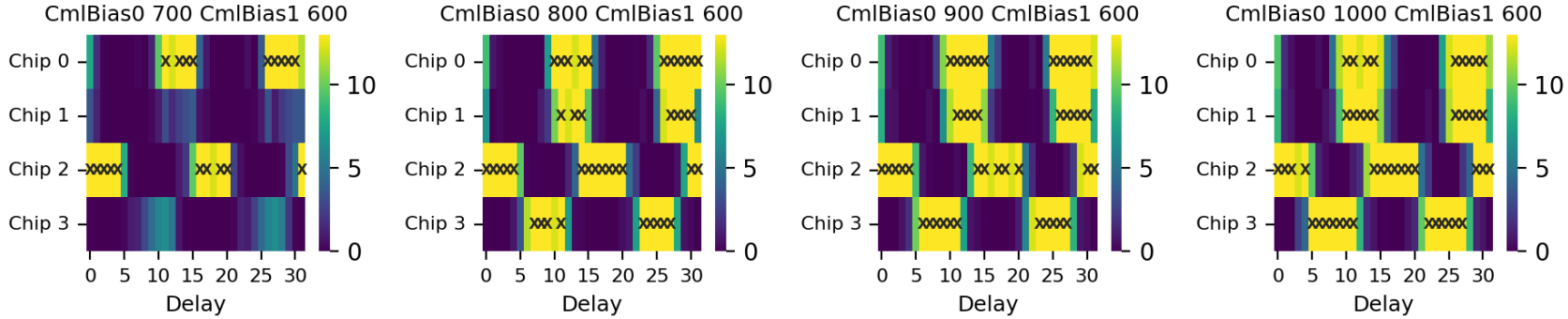


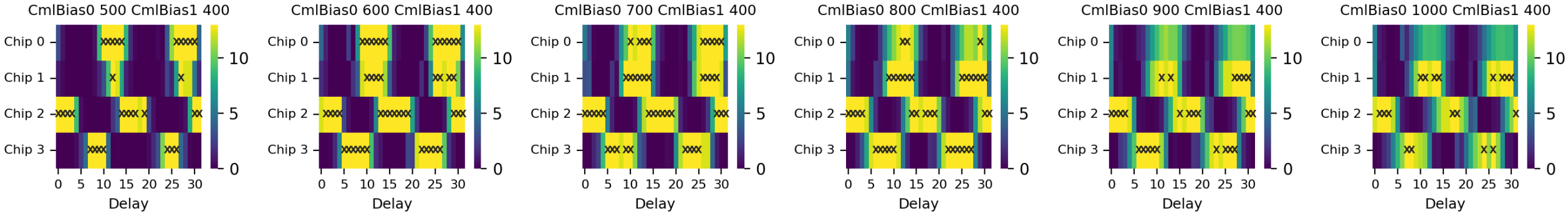
Nominal: 2.5 V, 1.28 Gbps, DiffTerm, VMUX 32

Trenz R2

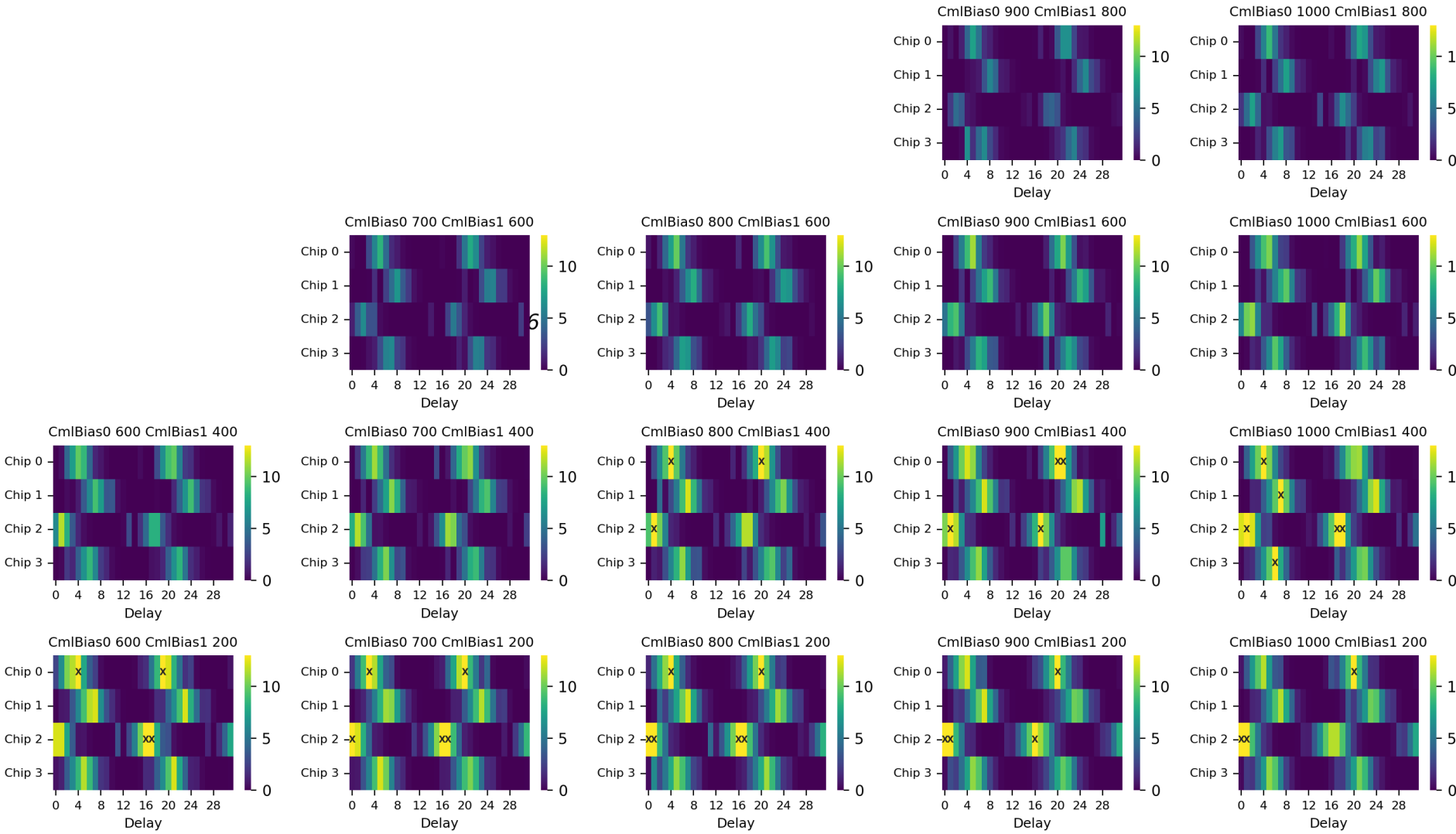
20UPGR92201045







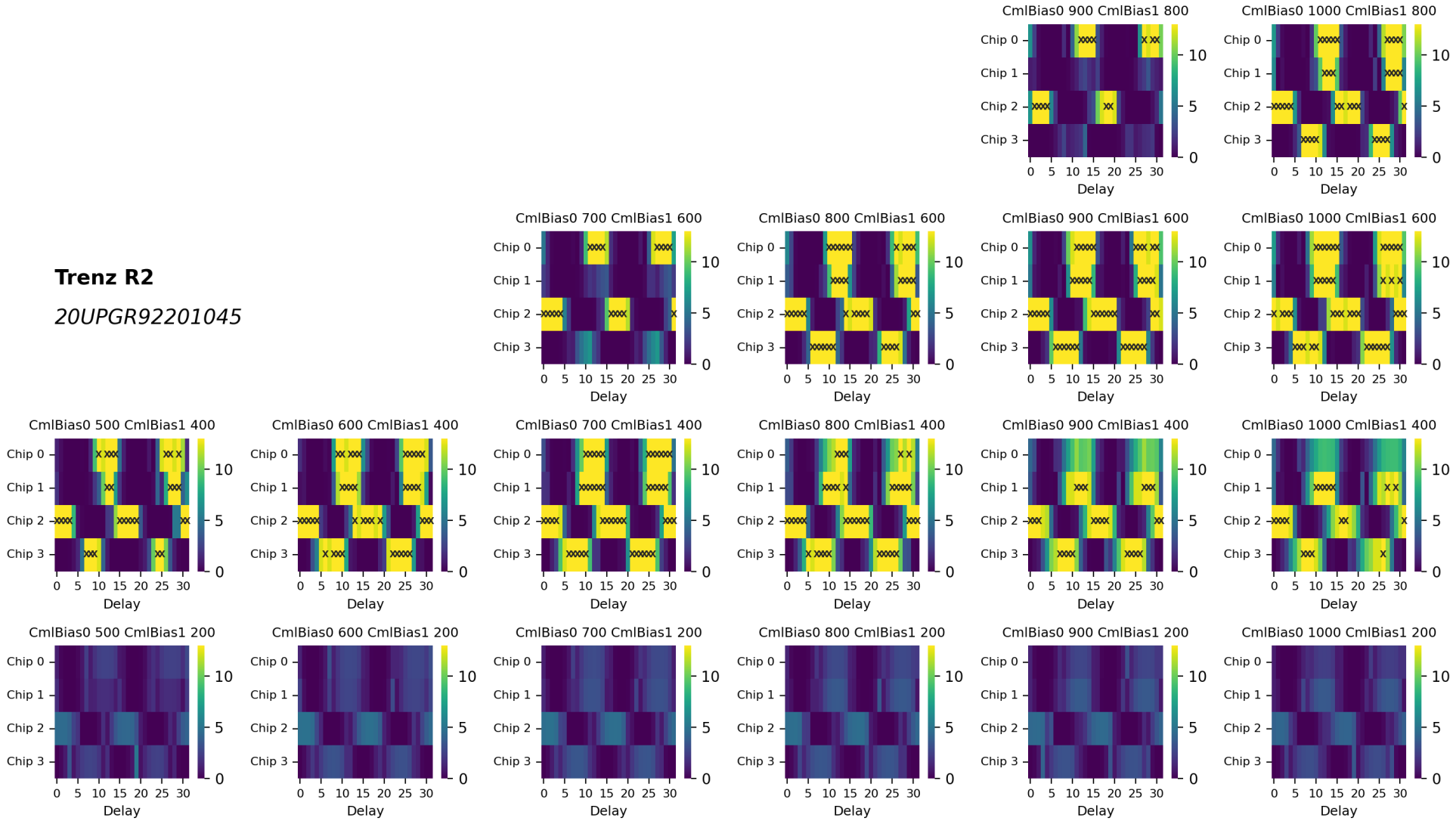
FPGA dependence: XpressK7



FPGA dependence: Trenz R2

Trenz R2

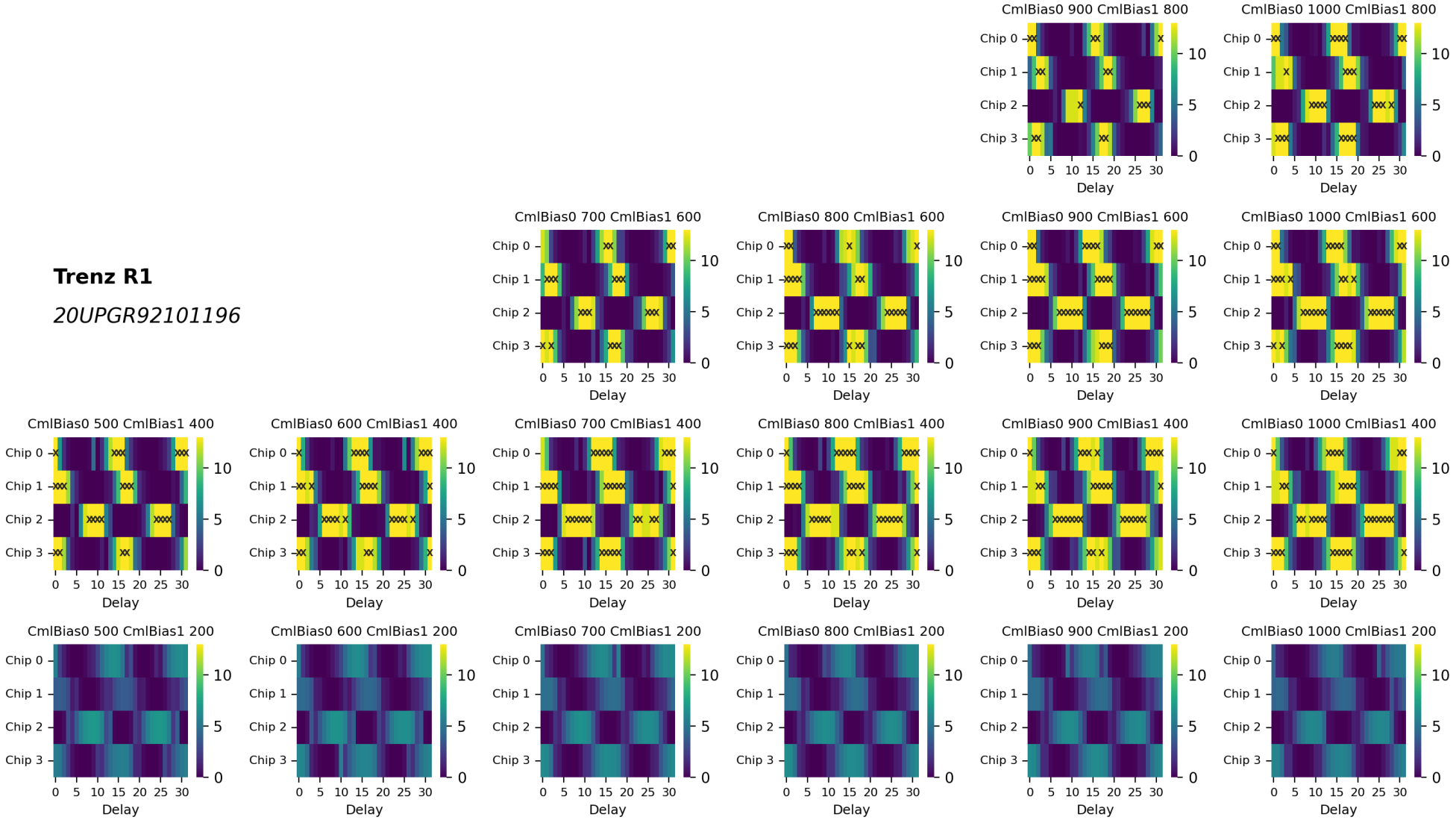
20UPGR92201045



FPGA dependence: Trenz R1

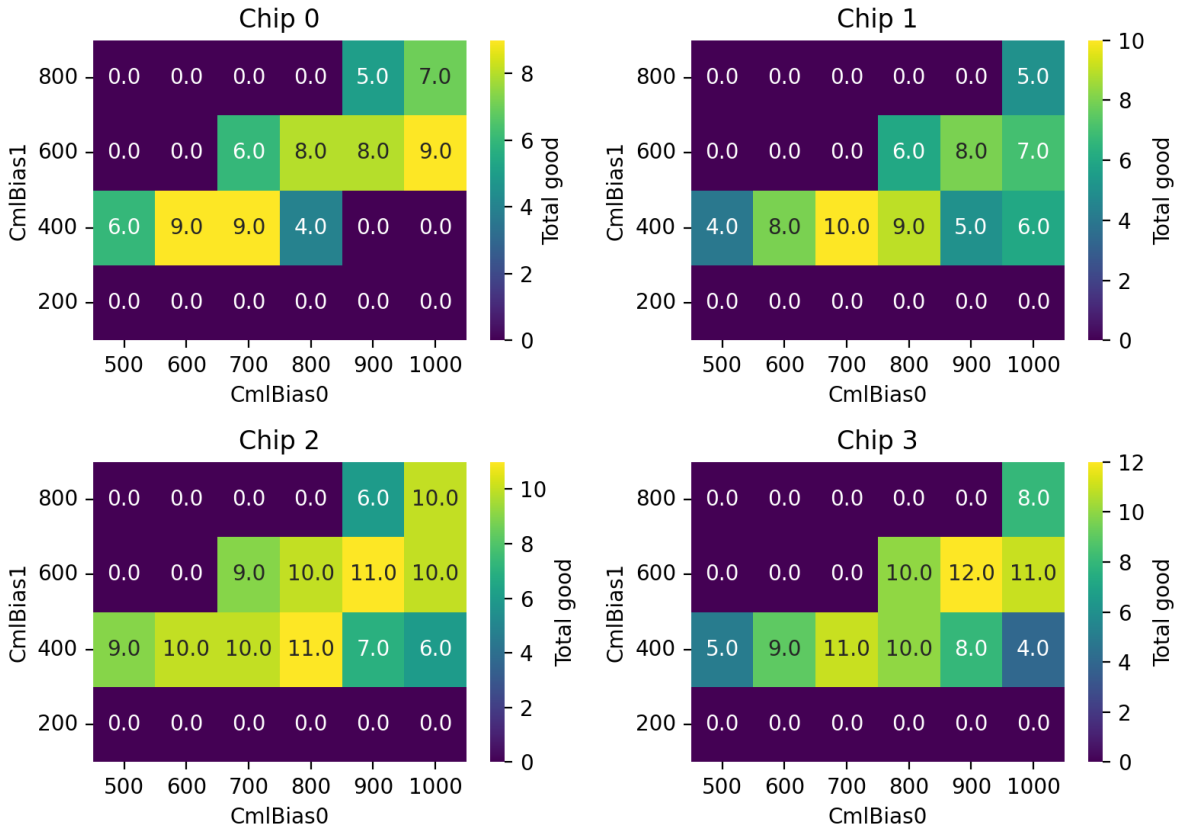
Trenz R1

20UPGR92101196

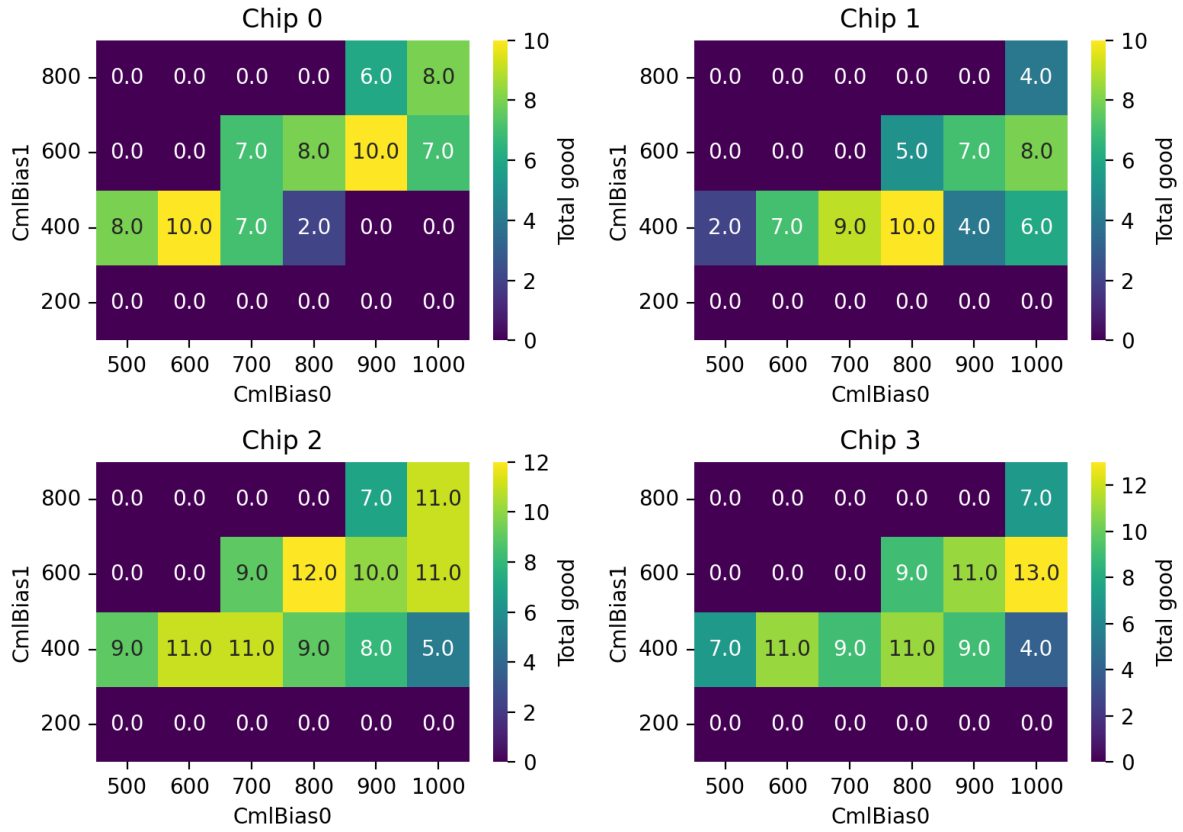


1.8 V vs 2.5 V

1.8 V

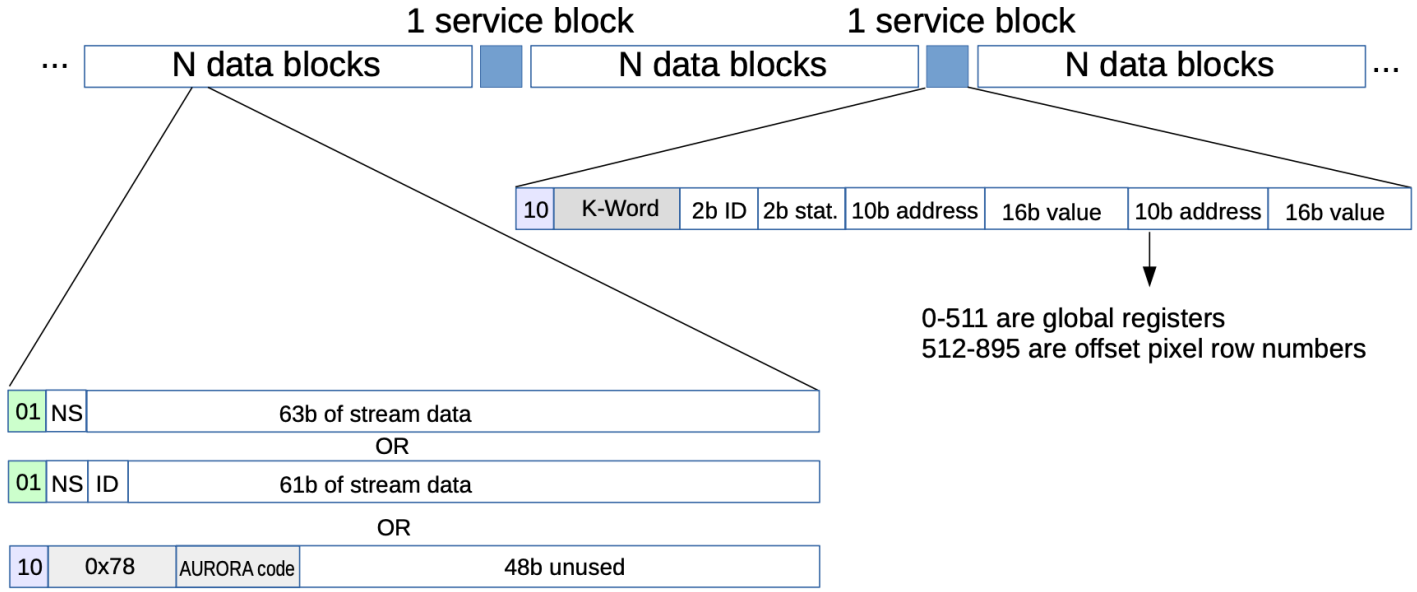


2.5 V



→ 2.5 V is maybe slightly better, but no significant difference, should probably stick with 1.8 V, as we've always done

Introduction



- RD53B uses Aurora64b66b protocol for data transmission
- Each Aurora block consists of 66 bits, with a 2-bit sync header (01 for data, 10 for service), plus 64 bits of scrambled data (scrambling means that the data stream is balanced, i.e. consists of an equal number of 0 and 1)
- Sync header block allows for frame alignment, i.e. identifying where each 66-bit block starts

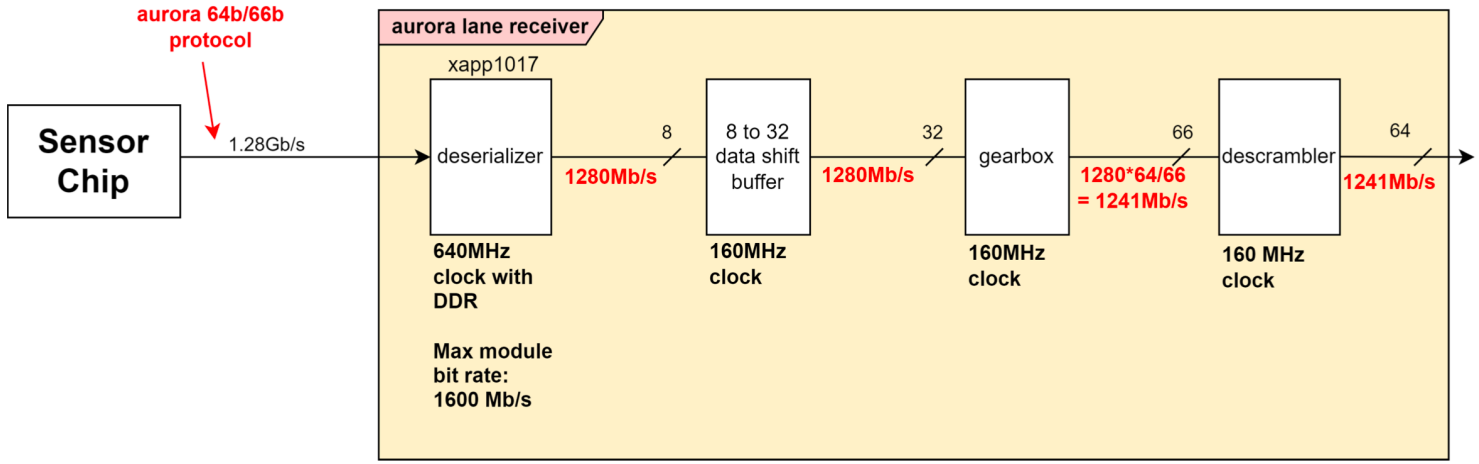
Data processing in YARR

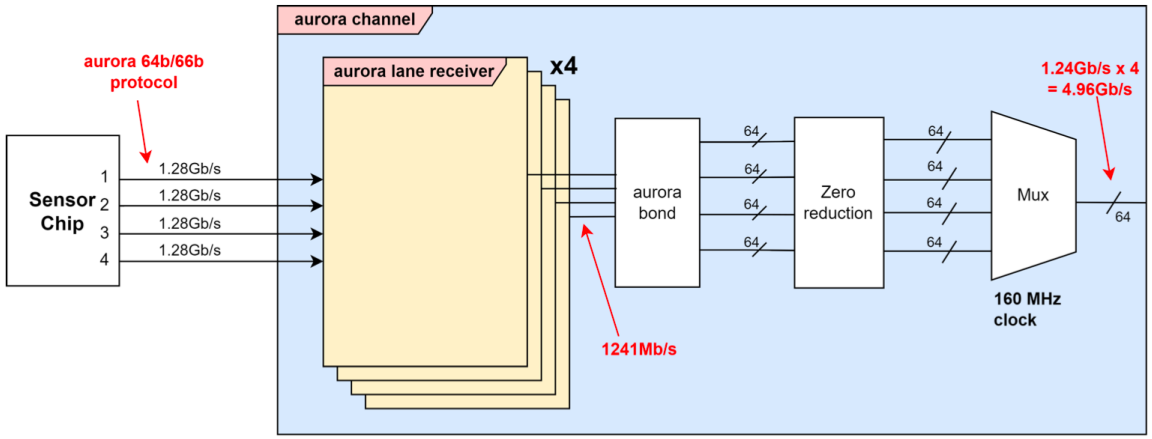
- Data transmission works via Aurora lanes:

1. Aurora lane receives aurora data blocks, which go into a deserializer
   - → Makes single bit signal into 8-bit signal
   - → **Previously deserializer was specific Xilinx application, which was trying to figure out sampling delay in a smart way**
   - → **With the new firmware, sampling delay is configurable from software**
2. Data goes into a 8 to 32 data shift buffer
3. 32 data goes into an internal buffer, which the gearbox process shifts through (in steps of 2) until it finds the sync blocks
4. Descrambler then extracts the 64-bits of data

- Several Aurora lanes can make up one Aurora channel, depending on what configuration we want, e.g.:
  - 4x4: 4 channels with 4 lanes
  - 16x1: 16 channels with 1 lane

Plots from Loic's thesis

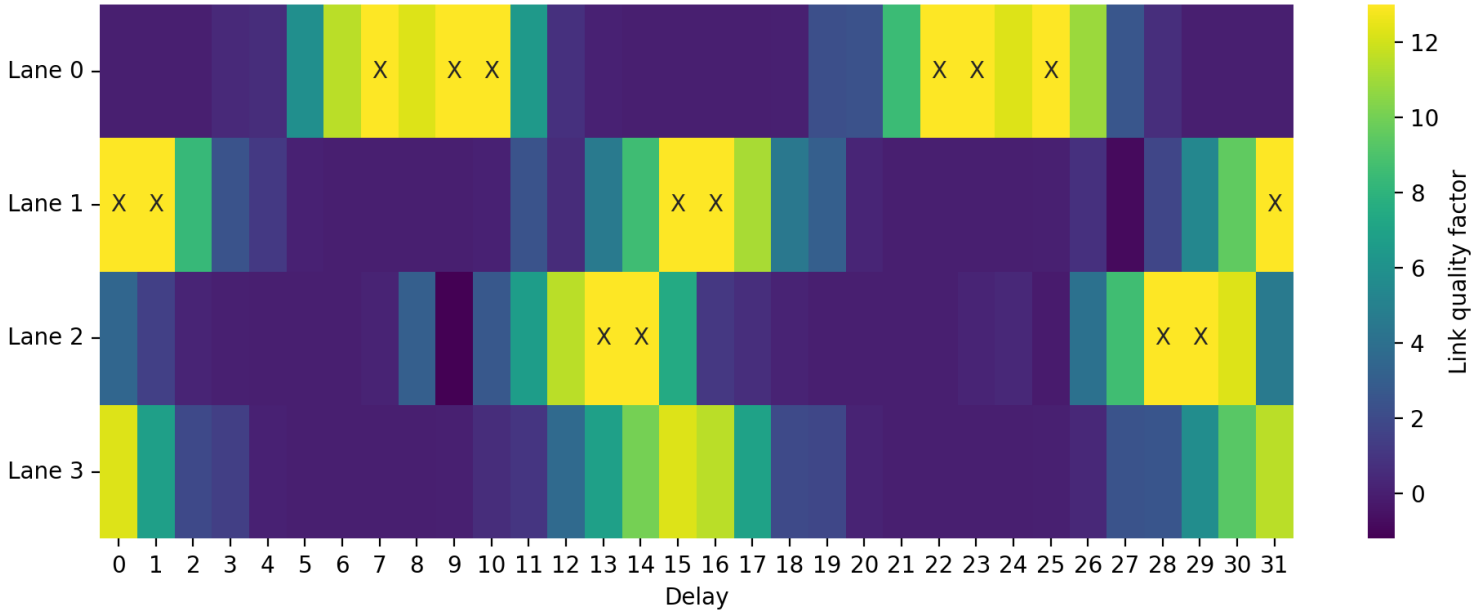




1.28 Gbps Firmware

- Was able to establish good communication on all lanes (and run clean digital scan), apart from 3
- Started playing around with VDDD and noticed that was accidentally running at quite high VDDD, and reducing VDDD makes the situation a lot worse

→ Timon suggested this points to an issue that was previously observed, where noise on Vref_PRE impacted the data transmission quality

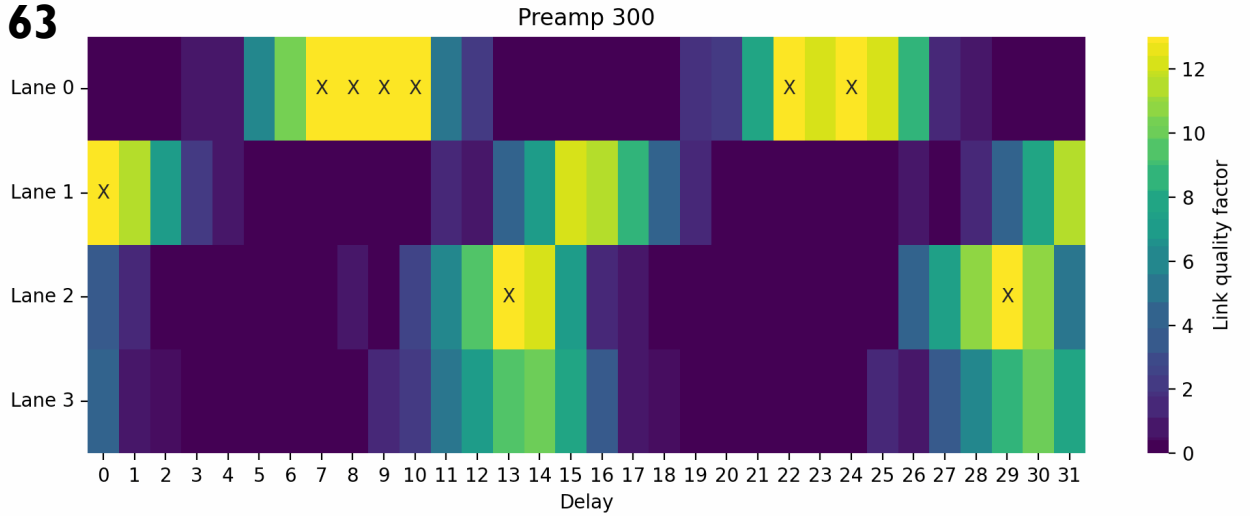


1.28 Gbps Firmware

- Previously it was seen that this behaviour improves with increasing the Preamp, or reducing the input current
- Both effects reproducible
- Also possible to mitigate this effect by setting VMUX=Vref_PRE

→ In this configuration I can get good data transmission for all lanes and a clean and stable digital scan

VMUX 63



VMUX 32